MODELING AND SIMULATION OF ETHANOL OXIDATION IN A CATALYTIC PACKED BED REACTOR USING DETAILED MULTICOMPONENT MODEL

SITI KHADIJAH BINTI SHAMSUL

CHEMICAL ENGINEERING
UNIVERSITI TEKNOLOGI PETRONAS

DECEMBER 2009

Modeling and Simulation of Ethylene Oxidation in a Catalytic Fixed Bed Reactor Using COMSOL Multiphysics -Ann Method

by

Siti Khadijah Binti Sukran

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Chemical Engineering)

DECEMBER 2009

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Modeling and Simulation of Ethylene Oxidation in a Catalytic Fixed Bed Reactor Using COMSOL Multiphysics -Ann Method

by

Siti Khadijah Binti Sukran

A project dissertation submitted to the
Chemical Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(CHEMICAL ENGINEERING)

Approved by,

(Dr Lau Kok Keong)
Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

DECEMBER 2009

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

SITI KHADIJAH BINTI SUKRAN

ABSTRACT

The purpose of this report is to discuss the research done and basic understanding of the topic **Modeling and Simulation of Ethylene Oxidation in a Catalytic Fixed Bed Reactor Using COMSOL Multiphysics -Ann Method.** The objective of this project is to develop a neural network system for ethylene oxidation in a catalytic fixed bed reactor. The COMSOL and MATLAB are the software used for this simulation. The scope of study for this project is to model the catalytic fixed bed reactor using COMSOL and train the neural network by using MATLAB software. The network will be trained using data obtained from COMSOL which has been verified by experimental data so that it can predict as close as possible to the actual behavior of the reactor. Finally, the output of the trained neural network will be represented in Graphical User Interface (GUI).

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The title of this project is, 'Modeling and Simulation of Ethylene Oxidation in a Catalytic Fixed Bed Reactor Using COMSOL Multiphysics -Ann Method'. In this study, a catalytic fixed bed reactor will be modeled using COMSOL. The data obtained will be used in developing a neural network in order to predict the behavior of ethylene oxidation inside the reactor. Prior to that, the data will be verified by experimental data. Two neural networks which are Feed Forward Backpropagation and NARX will be evaluated to identify the most suitable model to use. In completing the project, a GUI will be developed to illustrate the temperature profile, concentration profile and velocity profile of the modeled reactor.

1.1 Background of Study

The project aims is to use the application of COMSOL Multiphysics and artificial neural network (ANN) in modeling ethylene oxidation in a catalytic fixed bed reactor (CFBR). These two methods will permit process engineers to predict and solve desired design fluid dynamics in a reactor and any process equipment. COMSOL is a solver software package for various physics and engineering applications, especially multiphysics. It offers an extensive and well-managed interface to MATLAB and its toolboxes for a large variety of programming, preprocessing and postprocessing possibilities.

Artificial Neural Network (ANN) is the second tool that is used in this project. It is a technology that is being inspired by biological nervous system. The function of ANN is to solve problem that are difficult for conventional computers and human beings. The neural network will be trained to have the ability to recall and generalize from training pattern or set of data. Training the neural network will develop the ability to predict the output for a new set of input data.

1.2 Problem Statement

Fixed bed reactors (multitubular reactor) are commonly used for gas phase reactions, especially for oxidation reactions over solid catalyst. This is including the oxidation of ethylene to ethylene oxide. The characteristic features of a fixed catalyst are the pressure drop of the flowing gas in the catalyst bed and the danger of unstable operation points, especially with strongly exothermic reactions, when flow through the catalyst bed becomes nonuniform.

For that, conduct plant study is inappropriate method to predict and design the desired fluid dynamics in the fixed bed reactor. The computational simulation method is more viable, economical and time consuming compare to conduct plant experiment. Mathematical models are widely used to represent individual unit operations as well as entire processes accordingly to their behavior. However, the reliable representation of basic phenomena strongly relies upon model structure and the parameter values, which are to be adjusted, correspond to the real process and experimental data are required in order to estimate the model parameters to validate the developed model. It also leads to a non-linear dynamic partial differential equation which is difficult to solve.

To accomplish this project, the combined COMSOL and ANN method will be used to develop an intelligent neural network which is used to predict the operating condition inside FBR in a very short time as required by the industry. Finally, these profiles will be represented in a user-friendly Graphical User Interface (GUI).

1.3 Objective and Scope of Study

The main objective of this project is to develop a neural network system for ethylene oxidation in a fixed bed reactor using combined COMSOL and ANN method. To attain the main objective, there are several identified objectives that need to be achieved first.

- Modeling fixed bed reactor using COMSOL Multiphysics

- Simulate the data using neural network
- Develop graphical user interface (GUI)

The COMSOL Multiphysics and MATLAB are the software used for this modeling and simulation of fixed bed reactor. The scope of study for this project is to model the fixed bed reactor using COMSOL and develop the neural network using MATLAB software. Finally, the output of the trained neural network will be presented in Graphical User Interface (GUI).

CHAPTER 2

LITERATURE REVIEW

2.1 Ethylene oxide

Silver is a unique catalyst for the oxidation of ethylene to ethylene oxide. The reactions are as follows:

$$C_2H_4 + 0.5O_2 \Rightarrow C_2H_4O \qquad \Delta H = -117kJ/mol$$

$$C_2H_4 + 3O_2 \Rightarrow 2CO_2 + 2H_2O \qquad \Delta H = -1134kJ/mol$$

$$C_2H_4O + 2.5O_2 \Rightarrow 2CO_2 + 2H_2O \qquad \Delta H = -1217kJ/mol$$

The chief by-product is CO2. Ethylene is the only olefin which today can be converted selectively to the oxide (epoxide) by heterogeneous catalytic oxidation. Other olefins are oxidized over silver, but are converted mainly to CO2 as indicated in Table 1.

Table 1: Oxidation of Various Olefins over Silver Catalyst (W.G. Frankenburg)

Olefin	Rate	Products	Reference
C_2H_4	Medium	C_2H_4O; CO_2	(8,9)
C_3H_6	Medium	CO_2	(10,11)
1—C_4H_8	Medium-fast	CO_2	(12)
2—C_4H_8	Medium-fast	CO_2	(12)
i—C_4H_8	Fast	CO_2; trace acetone	(11,12)
C_4H_6	Medium-fast	CO_2	(12)

This is evidence of different adsorption of the olefins, and for reaction between adsorbed olefin and adsorbed oxygen rather than between gas olefin and adsorbed oxygen.

Ethylene oxide is made by direct reaction of ethylene and oxygen over silver catalyst. The reaction is exothermic, and the simultaneous even more exothermic oxidation of both ethylene and ethylene oxide leads to the byproducts carbon dioxide and water.

Ethylene oxide is one of the major individual compounds produced from ethylene. It is an important organic compound as it is an intermediate to other product such as ethylene glycol, surfactant and etc.

The production of ethylene oxide by the catalytic gas phase oxidation of ethylene with molecular oxygen in the presence of a silver catalyst has been widely practiced on the commercial scale. The catalyst beds consist of large bundles of many tubes that contain supported silver catalyst spheres or rings. The tubes are 6-12 meters long and 20-50 millimeters in diameter. The operating temperature for this reaction is at 200 to 250 °C. In the oxygen process, the reactor off-gas is fed to CO_2 scrubbers, then to ethylene oxide scrubbers which absorb the ethylene oxide into the liquid phase. The ethylene oxide is recovered from the liquid in a desorber and distilled to remove water. The reactor operates at a low conversion, however, so unreacted ethylene is fed to a secondary fixed bed reactor, then separated.

2.2 Fixed bed reactor

Reactors with a fixed catalyst bed are distinguished from those with moving catalyst. The characteristic features of a reactor with fixed catalyst are pressure drop of following gas in the catalyst bed. Fixed bed reactor must be shut down after a certain time onstream to generate or replace catalyst. Fixed bed reactor can be classified by the type of temperature control:

1) Reactor with no special temperature control features (adiabatic operation)
2) Reactor systems with stagewise temperature control
3) Reactors with continuous heat exchange along the flow path

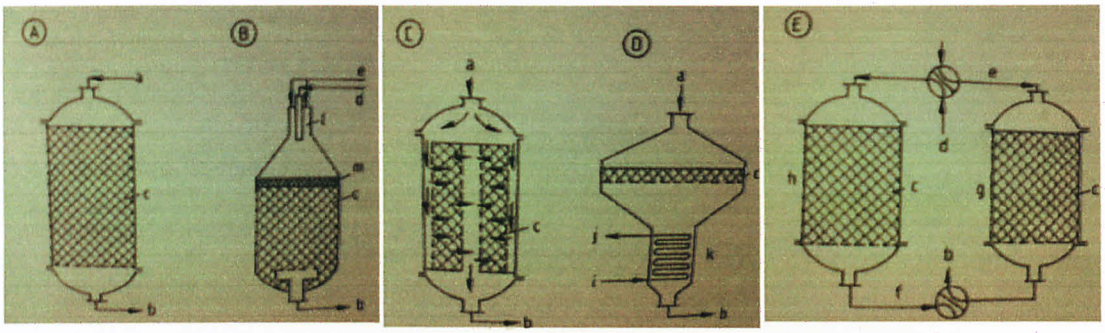


Figure 1: Fixed bed catalytic reactors for gas-phase reactions with no special provisions for temperature control (Ullmann's, 2005)

Figure A above shows Simple Fixed-Bed Reactor. This reactor is simple in design and not suitable for reaction with large heat of reaction and high temperature. It is commonly used in CO converting and amination of methanol to methylamines. Figure B shows Fixed Bed Reactor with Combustion Zone. This reactor operates using direct heating by combustion of part of hydrocarbon feed. It is commonly used in methane cleavage in secondary reformer. Figure C shows Radial Flow Reactor, which has lower pressure drop than axial-flow reactor. It is commonly used in ammonia synthesis. Figure D shows Shallow Bed Reactor, which used for high reaction rates and unstable products. It is commonly used in oxidation of ammonia to NO_x. Figure E shows Regenerative Furnace. It is very suitable when catalyst ages rapidly and can be regenerated. It is commonly used in dehydrogenation of butane to butadiene (Ullmann's, 2005).

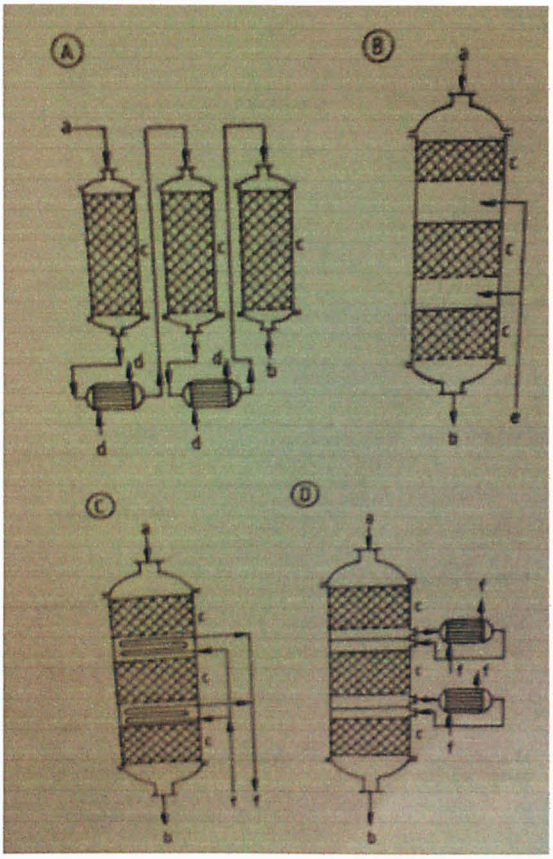


Figure 2: Fixed-bed catalytic reactors for gas-phase reactions with stagewise temperature control (Ullmann's, 2005)

Figure A above shows Cascade of Simple Fixed Reactor. It is used for large pressure and temperature differences are possible. This reactor commonly used in reforming heavy gasoline and hydrocracking. Figure B shows Multibed Reactor with Cold-Gas Injection. It is used for exothermic equilibrium reactions and commonly used in ammonia synthesis. Figure C shows Multibed Reactor with Interstage Cooling. It is also use for exothermic equilibrium reactions and in ammonia synthesis. Figure C shows Multibes Reactor with Heat Supply. It is used for endothermic equilibrium reactions and commonly used in dehydrogenation of ethylbenzene to styrene (Ullmann's, 2005).

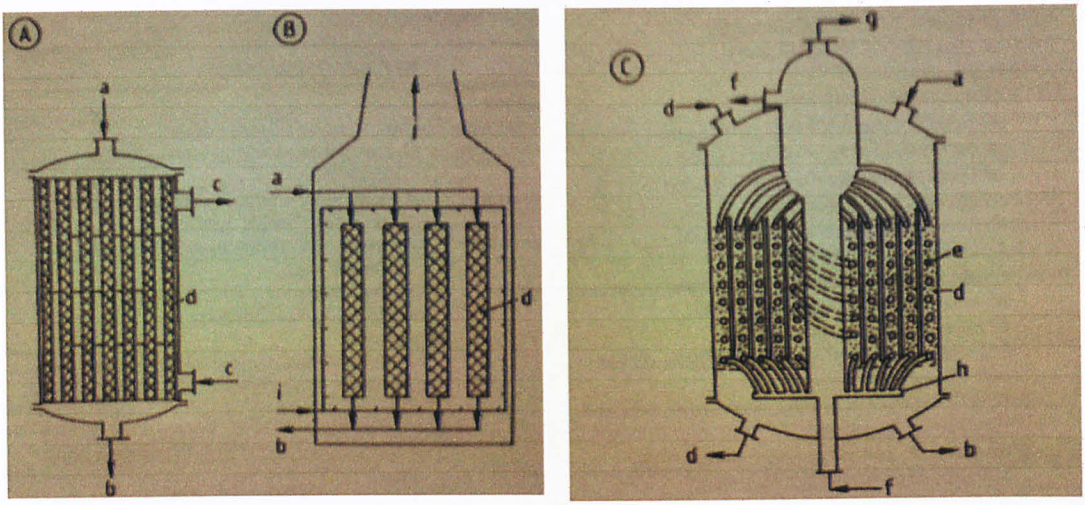


Figure 3: Fixed Bed catalytic reactors for gas phase reaction with continuous temperature control (Ullmann's, 2005)

Figure A shows Multitubular Reactor. It has temperature control with liquids, gaseous or boiling heat transfer agent in shell side space. It is relatively expensive design and has long downtime for catalyst replacement, usually twice a year. The internals in shell side space improves the heat transfer in the reactor. It is suitable for large pressure drop processing and most common used in industry i.e.: oxidation, alkylation, hydrogenation, dehydrogenation, hydration, dehydration, production of vinyl acetate from acetic acid, production of MTBE from methanol and isobutene, hydrochloronation including oxidation of ethylene to ethylene oxide (Ullmann's, 2005).

Figure B shows Tubular Furnace, which used direct heating of catalyst tubes for endothermic reactions and high reaction temperatures. It is commonly used in primary reformer. Lastly, Figure C shows Fixed Bed Reactor with heating and cooling element. It has advantages compared to other type of reactor when only the heating of cooling system has to be designed for special pressure condition. This reactor is suitable for methanol synthesis process (Ullmann's, 2005).

2.3 Navier Stokes Application Mode

Navier-Stokes equation is one of the equations used in modeling this simulation (Incompressible Navier-Stokes Model). In this project, Navier Stokes equation combined with Brinkman equation is applied for the porous region (packed bed that contains silver catalyst). Navier-Stokes equation is a basic equation describing movements of a fluid. It is a differential equation that describes the changes in momentum of an infinitesimal volume of fluid as the sum of friction, pressure changes, gravity and other forces acting inside the fluid as below. It assumed that momentum of a fluid must be conserved unless forces are acting on the fluid.

This equation describes flow in viscous fluid through momentum balances for each of the components of the momentum vector in all spatial dimensions. They assume the density and viscosity of the modeled fluid are constant, which gives rise to a continuity condition. The model of this equation is able to account for arbitrary variations in density. The momentum balance and continuity equation form a non-linear system of equation with three or four coupled equations in 2D and 3D respectively. The equation is as follows:

$$\rho\frac{\partial u}{\partial t} - \nabla\eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}}) + \rho\mathbf{u}.\nabla\mathbf{u} + \nabla p = \mathbf{F}$$

$$\nabla\mathbf{u} = 0$$

η = dynamic viscosity ($ML^{-1}T^{-1}$)

$\mathbf{u}$ = *velocity vector* (LT^{-1})

ρ = *density of the fluid* (ML^{-3})

$\mathbf{p}$ = *pressure* ($ML^{-1}T^{-2}$)

$\mathbf{F}$ = *body force term* ($ML^{-2}T^{-2}$)

There are many boundary conditions offer the modeling. One of it is Inflow/Outflow boundary condition (for an imposed velocity) as below:

$$\mathbf{u}.\mathbf{n} = u_0$$

To impose a certain pressure in the Outflow/Pressure boundary condition:

$$p=p_0$$

There is special boundary condition that combines both these description which is called Normal Flow/Pressure boundary condition. This condition sets the velocity component in the tangential direction to zero and sets the pressure to specific value. It is available to simulate channels that are long in length and where flow is assumed to have stabilized so that all velocity occurring in the tangential direction is negligible. Imposing this condition removes the necessity of simulating this length.

$$\mathbf{u}.\mathbf{t}=0 \text{ and } p=p_0$$

The slip symmetry condition states that there is no velocity component perpendicular to a boundary:

$$\mathbf{u}.\mathbf{n}=0$$

The Neutral boundary condition states that transport by shear stresses is zero across a boundary. This boundary condition is denoted neutral since it does not put any constraints on the velocity and stats that there are no interactions across the modeled boundary.

$$\eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}})\mathbf{n} = 0$$

No slip boundary condition eliminates all components of velocity vector:

$$\mathbf{u} = 0$$

The laminar inflow/outflow boundary condition is appropriate for low Reynolds number flow regimes, where a fully developed laminar profile is expected.

$$L_{inl}\,\nabla_t.[p\mathbf{I}-\eta(\nabla_t\mathbf{u}+(\nabla_t\mathbf{u})^T)]-\mathbf{n}p_{inl}$$
$$\nabla_t.\mathbf{u}=0$$

In the above equation, L_{inl} gives the length of inlet channel and p_{inl} the pressure at the end of the inlet channel.

2.4 Convection and Conduction Application Mode

The general equations for an energy balance, which considers heat transfer through convection and conduction, in the Chemical Engineering module is:

$$\rho C\frac{\partial T}{\partial t}+\nabla.(-k\nabla T+\rho C_p T_{\mathbf{u}})=Q$$

Cp=specific heat capacity ($L^2\,T^{-2}\theta^{-1}$) T= Temperature (θ)

k= thermal conductivity ($MLT^{-3}\,\theta^{-1}$) **u**= velocity vector (LT^{-1})

Q= sink/ source term ($ML^{-1}T^{-3}$)

Chemical reactions that often arise in energy balances can be accounted for by arbitrary reaction mechanisms, introduced as heat sink/ heat sources. The equation above includes velocity vector, **u**, which can be expressed analytically or obtained by coupling a momentum balance. If the turbulent model property has the value of k-epsilon, k is replaces by ($k+k_T$) where k_T is turbulent heat conductivity.

The available boundary conditions in the Convection anad Conduction application mode include an imposed Heat flux condition:

$$\mathbf{q}.\mathbf{n}=q_0$$

where **q** denotes the heat flux vector ($ML^{-1}T^{-3}$) and is defined by the expression within parentheses in above equation, namely:

$$\mathbf{q} = -k\nabla T + \rho Cp T\mathbf{u}$$

The source term q0 can be arbitrarily defined to represent flux into an inmate medium, a radiation term, and heat from chemical reactions or nuclear decay. The thermal insulation condition has the source term as below:

$$\mathbf{q}.\mathbf{n} = 0$$

This boundary condition can also be used as symmetry conditions. To impose the temperature condition, following boundary condition can be used:

$$T=To$$

The final boundary condition, denoted convective flux, assumes that all energy passing through this boundary does so through the convective flux mechanism. This firstly assumes that any heat flux due to conduction across this boundary is zero:

$$\mathbf{q} = -k\nabla T.\mathbf{n} = 0$$

So that the resulting equation becomes:

$$\mathbf{q}.\mathbf{n}=\rho CpT\mathbf{u}.\mathbf{n}$$

This is a useful boundary condition, particularly in convection-dominated energy balances where the outlet temperature is unknown.

2.5 Convection and Diffusion Application Mode

The convection and diffusion equation is:

$$\frac{\partial c_i}{\partial t} + \nabla.(-D_i \nabla c_i + c_i \mathbf{u}) = R_i$$

ci= concentration of species (ML^{-3}) Di= diffusion coefficient (LT^{-1})

u= velocity vector (LT^{-1}) Ri= reaction term (ML^3T^{-1})

In the reaction term, arbitrary kinetic expressions of reactants and products can be introduces. The expression within the brackets represents the flux vector, where the first term describes transport by diffusion and the second represents the convective flux:

$$\mathbf{N}_i = -D_i \nabla c_i + c_i \mathbf{u}$$

Where Ni= mass flux vector ($ML^{-2}T^{-1}$).

The diffusion coefficient for the dissolves species accounts for interaction between solute and solvents. In the Maxwell-Stefan convection-diffusion application mode, the interaction between the different dissolved species is also taken into account. The available boundary condition includes an imposed of Mass flux condition:

$$\mathbf{N_i}.\mathbf{n}=N_0$$

where the boundary source term, N_0, can be arbitrarily defined to represent flux into an infinite medium or material change, most often through chemical reaction. The insulation/ symmetry condition has the source term which as below:

$$\mathbf{N_i}.\mathbf{n} = 0$$

The boundary condition to impose concentration also as below:

$$c_i = c_{i,0}$$

The Convective flux boundary condition assumes that all mass passing through this boundary is convection-dominated. This firstly assumes that any mass flux due to diffusion across this boundary is zero:

$$\mathbf{n}.(-D_i \nabla c_i) = 0$$

So that, $\mathbf{N}_i.\mathbf{n} = c_i.\mathbf{u}.\mathbf{n}$

This a useful boundary condition, particularly in convection-dominated mass balances where the outlet concentration is unknown.

2.6 COMSOL

The basic numerical method used in COMSOL is the so-called finite-element method (FEM). COMSOL is design to solve a wide range of partial differential equations comprising most of the equation appearing in physics and chemistry such as the Navier-Stokes equation, the Poisson equation, the Schrodinger equation, the convection-diffusion equation, the Maxwell equations and problem combining these equations. COMSOL allows unexperience user to quickly get started and solve fairly complex micsrofluidics problems, while it still remain powerful tool for experienced user.

2.7 Neural Networks

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. Neural network can be trained to perform a particular function by adjusting the values of the connections (weights) between elements.

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are needed to train a network.

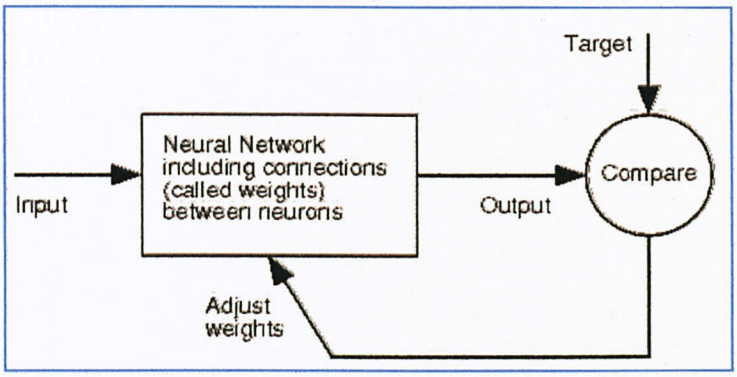


Figure 4: Neural Network Concept

Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems.

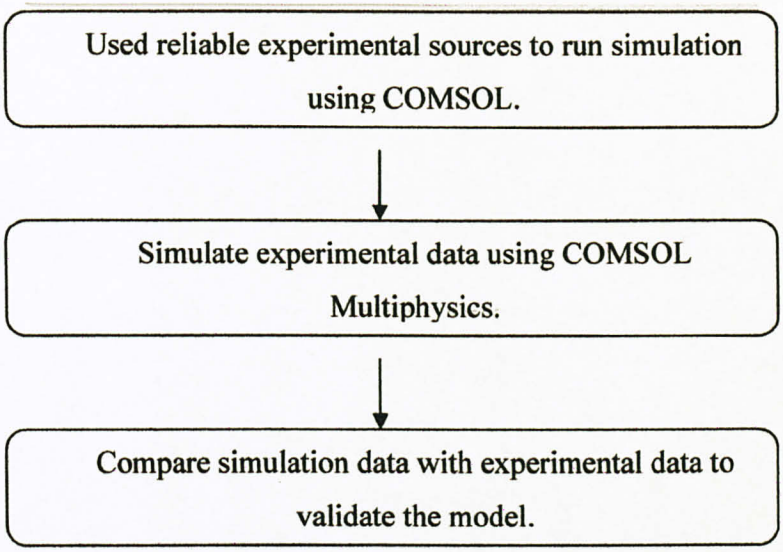
Today neural networks can be trained to solve problems that are difficult for conventional computers or human beings. Throughout the toolbox emphasis is placed on neural network paradigms that build up to or are themselves used in engineering, financial, and other practical applications.

CHAPTER 3

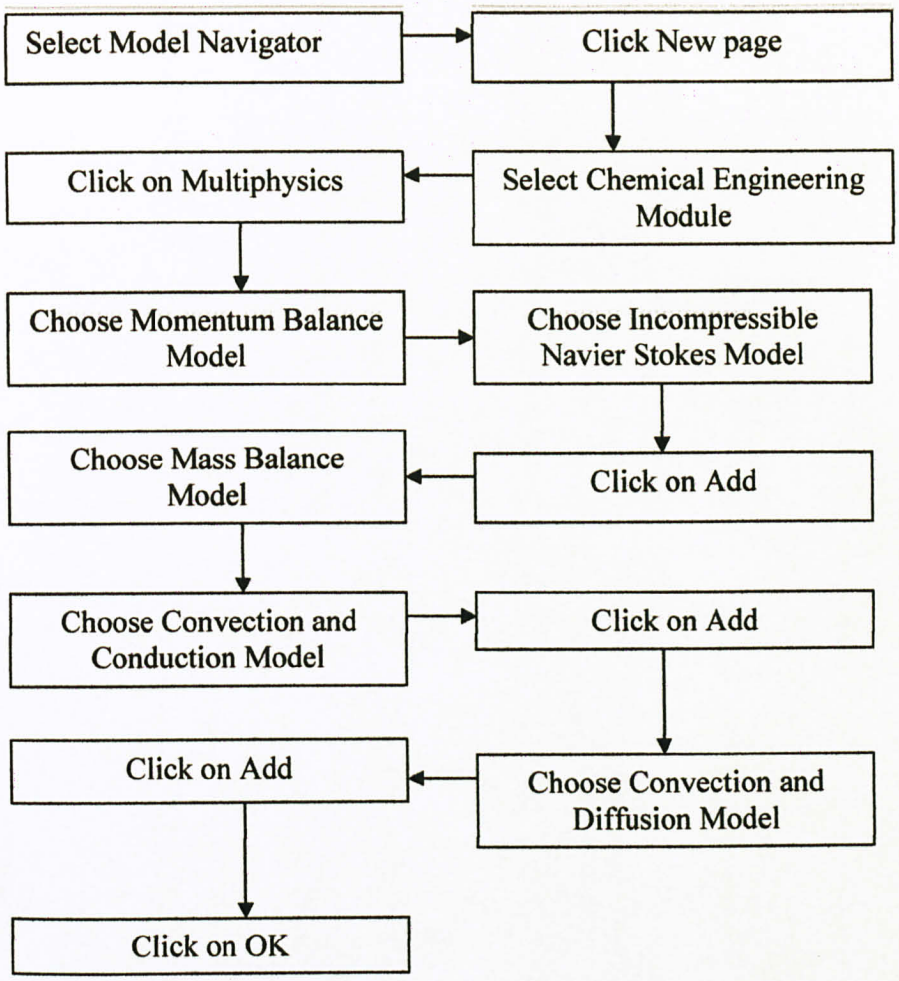
METHODOLOGY

3.1 Flowchart of project work flow

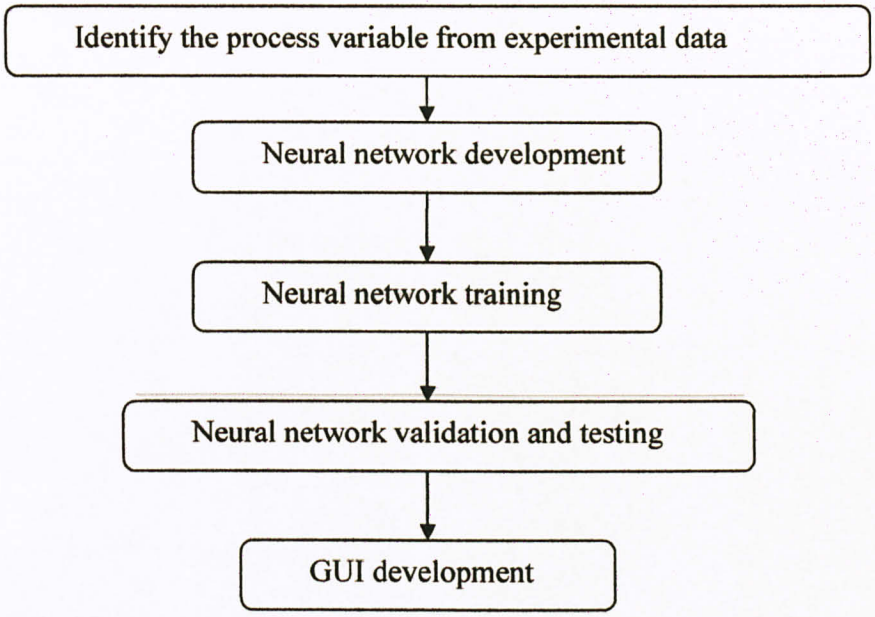
Part 1: COMSOL simulation



3.1.1 Starting COMSOL-Multiphysics



Part 2: Neural Network training



3.2 Gantt Chart

No.	Detail/ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Project work ontinue															
2	Submission of Progress Report 1				x											
3	Project Work Continue															
4	Submission of Progress Report 2									x						
5	Seminar															
6	Project work continue															
7	Poster Exhibition												x	x		
8	Submission of Dissertation (soft bound)														x	
9	Oral Presentation															x
10	Submission of Project Dissertation (Hard Bound)															x

CHAPTER 4

RESULT AND DISCUSSION

Part 1: COMSOL simulation

4.1 Operating profiles

4.1.1 Incompressible Navier-Stokes model

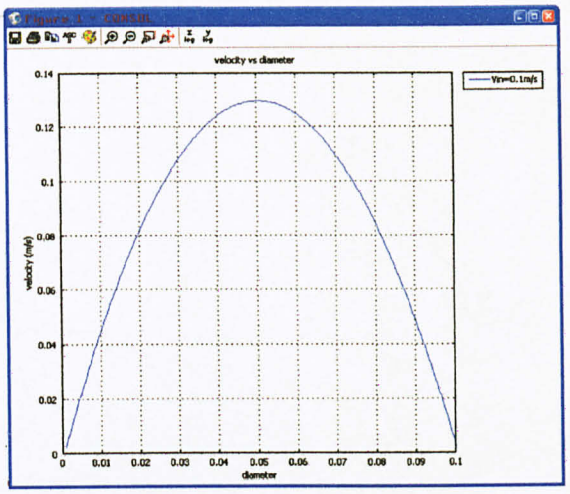


Figure 5: Velocity profile of moving fluids

Figure above illustrates the velocity profile with respect to diameter of reactor column. The diameter of this column is 0.1 m. The velocity of stream is varies along the diameter of the column. As the fluids moving toward the center of the column, the velocity of the fluid is increasing meanwhile, the velocity of the moving fluids decreasing towards the wall region for both sides. Based on this profile, the maximum velocity is 0.13 m/s which happened to occur at the center of the column. In contrast, the minimum velocity of stream occurs at the wall of the column. These phenomena occur due to friction that exists at the wall with the moving fluids that cause the flow of fluids near the wall to reduce. The molecules near to the wall have high friction compare to the

molecules far from the wall. This will result high velocity at the center of reactor column and low velocity near the wall.

4.1.2 Convection and Conduction Model

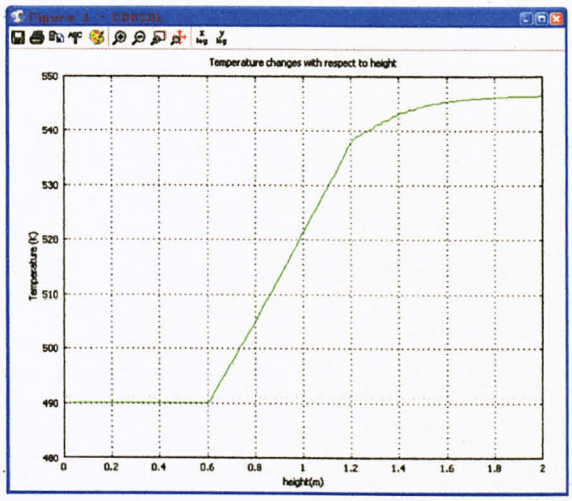


Figure 6: Temperature profile of ethylene oxidation process

Figure above illustrate the temperature changes along the height of the column. The height of the column is 2m. The temperature is constant at 490 K from 0 to 0.6 meter and start to increase after 0.6 meter high. In other words, the temperature starts to increase toward the outlet of the reactor. The increase in temperature is due to the heat of reaction that has been release in this process. Ethylene oxidation occur under exothermic reaction, therefore the temperature outlet of this reactor must be higher compare to the temperature inlet of the reactor. During this process ethylene molecules are absorbing the heat to break the bond and react with oxygen in order to produce ethylene oxide.

4.1.3 Convection and Diffusion Model

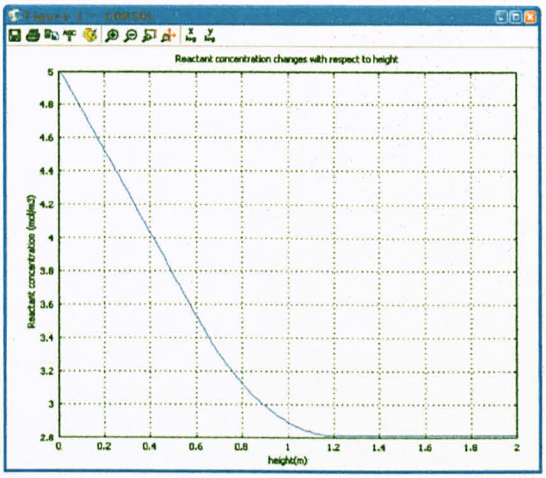


Figure 7: Concentration profile of ethylene with respect to the height.

From above figure, initial concentration of ethylene at the inlet of the reactor is 5mol/m3. Towards the outlet of the reactor, ethylene concentration starts to decrease until it reached 2.8mol/m3. In this reaction, the limiting reactant is oxygen, therefore when the oxygen has been reacted completely; it leaves ethylene as the excess reactant. The reduction of ethylene concentration is reasonable because ethylene has been converted to the ethylene oxide during the oxidation process.

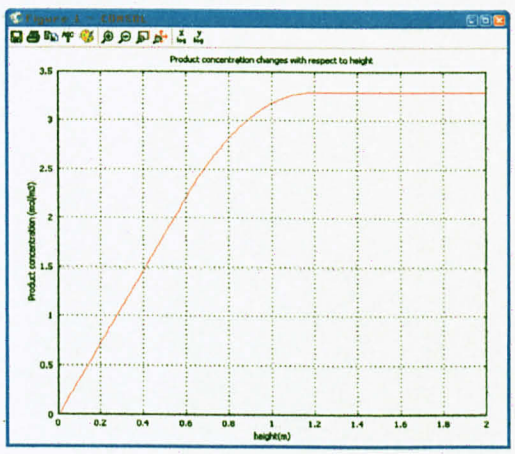


Figure 8: Concentration profile of ethylene oxide with respect to the height.

The above graph shows the concentration profile of ethylene oxide product in the reactor with respect to the height of the column. The product concentration at the inlet of reactor is zero. This is because there is no product has been produce at the inlet of reactor yet. The ethylene oxide concentration is increasing towards the reactor outlet. The product has only been produce once the oxidation reaction takes place in the reactor. Therefore the product has high concentration at the outlet of reactor due to product accumulation at this region.

4.1.4 Effect of inlet temperature changes towards product's concentration

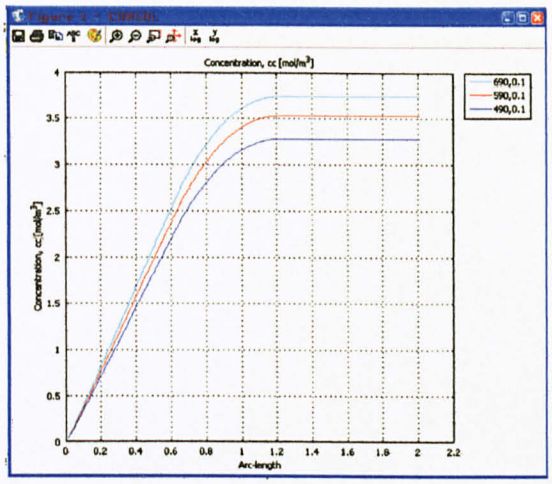


Figure 9: Concentration profile of ethylene oxide at different inlet temperature

The graph displays the effect of ethylene oxide concentration profile with different inlet temperatures while maintain other variables constant. At temperature 490 K, the maximum product concentration is 3.25 mol/m^3; at temperature 590 K, the maximum product concentration is 3.5 mol/m^3; while at 690 K the maximum product concentration is at 3.75 mol/m^3. Based on this profile, the higher inlet temperature will produce more ethylene oxide product. Collisions of particle will only result in a reaction if the particles collide with enough energy to get the reaction started. Thus, by increasing the temperature, the rate of reaction will increase and the numbers of energetic particles will also increasing, therefore produced more products.

4.2 Reactor column cross section area

Based on simulation had been done, the maximum diameter of cross section area is 0.81m. Referring to the journal, normal value of diameter column is between 0.5 – 1.0 m and process best operates at 0.8m. Below are the simulated models between the experimental size and the maximum simulated size of each interested operating parameters.

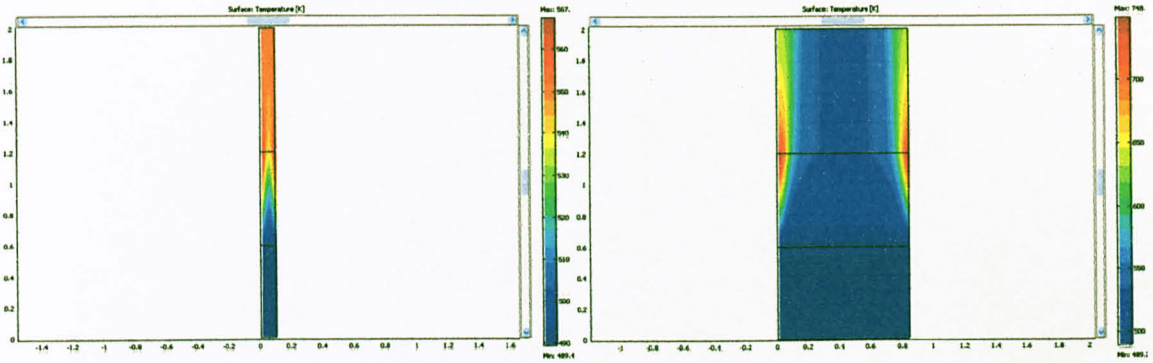


Figure 10: Temperature profile at cross section of 0.1m and 0.8m respectively

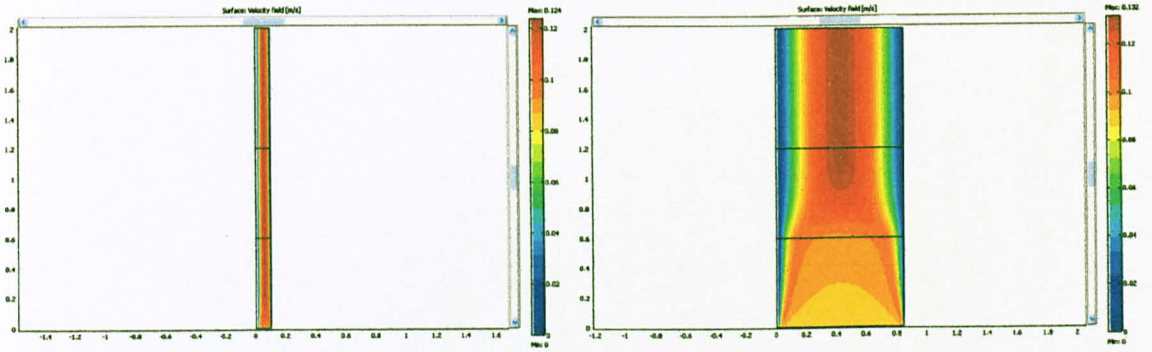


Figure 11: Velocity profile at cross section column of 0.1m and 0.81m respectively

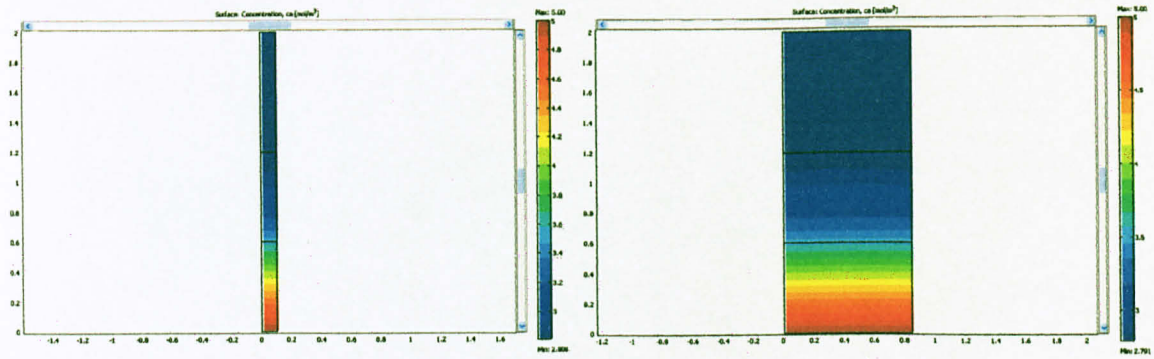


Figure 12: Reactant's concentration profile at cross section column of 0.1m and 0.81m respectively

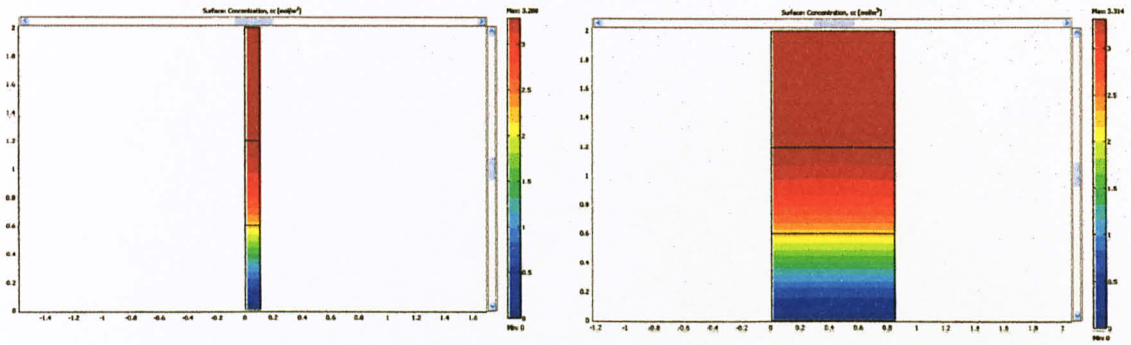


Figure 13: Product's concentration profile at cross section column of 0.1m and 0.81m respectively

4.4 MODEL'S VALIDATION

In order for the simulation data to be reliable, it is needed to be validated or compared with the experimental data. This project is based on an experimental data taken in a book, "*Experiments in Catalytic Reaction Engineering*", published by Elsevier Science, Netherlands. The simulation data is compared with the experimental data, obtained in this book, and hence the error of the data is analyzed. Table 1 is showing the error/ differences between simulation and experimental results

Figure 14: The book that been referred for experimental data

Table 2: Error calculation for the simulation results

RUN	Subdomain integration value,m^2K, (I)	Area,m2, (A)	Texperimental, K (Texp)	Tsimulation,K, I/A, (Tsim)	% Error 100(Tsim-Texp)/ Texp
1	31.48	0.06	525.2	524.67	0.06
2	32.02	0.06	535.8	533.73	0.39
3	32.64	0.06	545.2	544.06	0.21

In order to verify the simulation results, there are three simulation been done at different temperature. The simulation temperature for each simulated model is compared with the experimental value. As shown in the table, all of the results have less than 1% of error. As the error is very small, the simulation is considered to be reliable.

One of the model validations is as below:

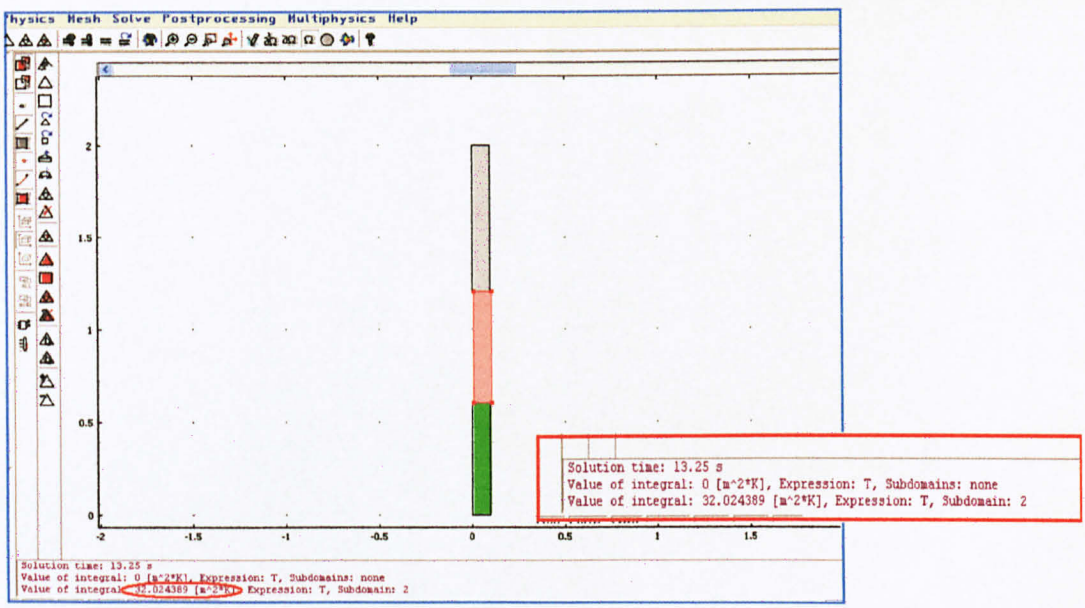


Figure 15: Subdomain integration value is $32.02m^2K$ at inlet temperature of 500K

Part 2: Neural Network Training

4.3 Identification of process variables

Specification of controlled variables, manipulated variables and disturbance variables is a critical step in developing a control system. It is the major part of neural network construction. There are three important type of process variable which are controlled variable, manipulated variable and disturbance variable.

- Control variables are the variables which quantify the performance or quality of final product, which are also called output variables. The control variables for this reactor are the outlet velocity, outlet temperature and outlet concentration.
- Manipulated variables are the variables that being adjusted dynamically to keep the controlled variables at their set-points. The manipulated for this reactor are the inlet velocity, inlet temperature and inlet concentration.
- Disturbance variables are also called 'load' variables and it represents input varia bles that can cause the controlled variables to deviate from their respective set-points.

As a conclusion, the overall process variables that involves in this project are fluid velocity, fluid temperature, fluid concentration and coordinate of fluid particle (x,y).

4.4 Neural Network Development

4.4.1 Feed Forward Backpropagation Network

There are generally four steps in the training process which are:

- Assemble the training data
- Create the network object
- Train the network
- Simulate the network response to new inputs

There are several transfer functions that can be associated with backpropagation but the most commonly used are log-sigmoid transfer function, tan-sigmoid transfer function and linear transfer function (H. Demuth & M. Beale, 1996).

A log sigmoid transfer function will generate output between 0 and 1 as the neuron's net input goes from negative to positive infinity. A tan-sigmoid transfer function will generate output between -1 and 1. Occasionally, the linear transfer function can generate a wide range of output.

The configuration of the feed forward is determined from trial and error. The number of layers, neurons and training function is changed for several times. Each configuration is compared to determine the best configuration. The best configuration is summarized in table below.

Table 3: Feed Forward Network Configuration

Parameters	Variable
Network	Feed forward backpropagation (newff)
Training function	TRAINLM
Adaptation learning function	LEARNGDM
Performance function	MSE
Epochs	100
Number of layers	3
Layer 1: Number of neuron **Transfer function (PURELIN)**	5
Layer 2: Number of neuron **Transfer function (PURELIN)**	34
Layer 3: Number of neuron **Transfer Function (PURELIN)**	1

The first step in training a feedforward network is to create the network object. The function `newff` creates a feedforward network. Before training a feedforward network, the weights and biases must be initialized. The `newff` command automatically initializes the weights. This command creates the network object and also initializes the weights and biases of the network, therefore the network is ready for training.

Once the network weights and biases are initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. The training process requires a set of proper network behavior, which is network inputs p and target outputs t. During training, the weights and biases of the network are iteratively adjusted to minimize the network performance function `net.performFcn`. The default performance function for feedforward networks is mean square error. Mean square error is the average squared error between the network outputs a and the target outputs t.

There are several training algorithms for feedforward networks. All these algorithms use the gradient of the performance function to determine how to adjust the weights to minimize performance. The gradient is determined using a technique called backpropagation, which involves performing computations backward through the network. Gradient descent and gradient descent with momentum are examples of training algorithm. However, these two methods are too slow for practical problems. Therefore it is preferable to use faster training algorithm to solve a complex problem.

The training algorithm that is being used in feedforward backpropagation in this project is Levenberg-Marquardt algorithm (`trainlm`). This training algorithm has the fastest convergence for networks that contain up to a few hundred weights. In many cases, `trainlm` is able to obtain lower mean square errors than any of the other algorithms tested. However, as the number of weights in the network increases, the advantage of `trainlm` decreases.

The network is trained based on above configuration. Post-regression analysis, RMSE and CDC are important characteristics to evaluate the network. The next section will discuss about post regression analysis.

4.4.1.1 Concentration Training using Feedforward Backpropagation

The Levenberg-Marquardt algorithm is used for training. The training stopped after 9 iterations because the validation error increased.

```
TRAINLM-calcjx, Epoch 0/100, MSE 30.4159/0, Gradient 87.6008/1e-010
TRAINLM-calcjx, Epoch 9/100, MSE 0.0592837/0, Gradient 8.67395e-010/1e-010
TRAINLM, Validation stop.
```

The result is shown in the following figure. The result here is reasonable, because the test set error and the validation set error have similar characteristics, and it does not appear that any significant over fitting has occurred.

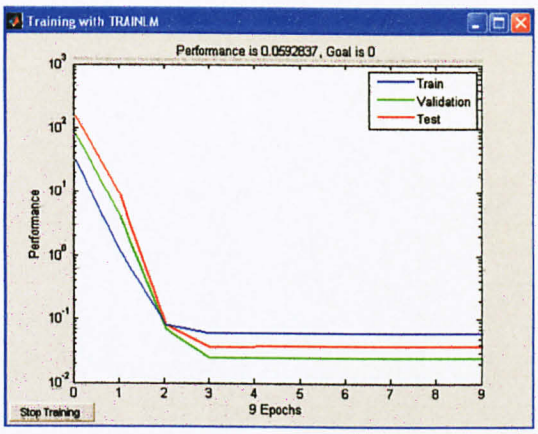


Figure 14: Concentration Training

4.4.1.1.1 Post-Training Analysis for Feedforward Backpropagation Network

The performance of a trained network can be measured to some extent by the errors on the training, validation, and test sets, but it is often useful to investigate the

network response in more detail. One option is to perform a regression analysis between the network response and the corresponding targets. The routine `postreg` is designed to perform this analysis.

If there were a perfect fit (outputs exactly equal to targets), the slope would be 1, and the y-intercept would be 0. The third variable returned by `postreg` is the correlation coefficient (R-value) between the outputs and targets. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is perfect correlation between targets and outputs. In this case, there are three outputs, so perform three regressions. The results are shown in the following figures.

a. Testing

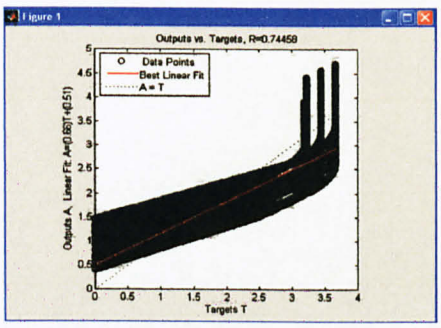


Figure 15: Post Regression for Concentration Testing

b. Validation

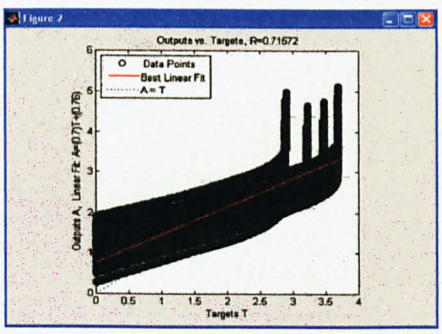


Figure 16: Post Regression for Concentration Validation

c. Training

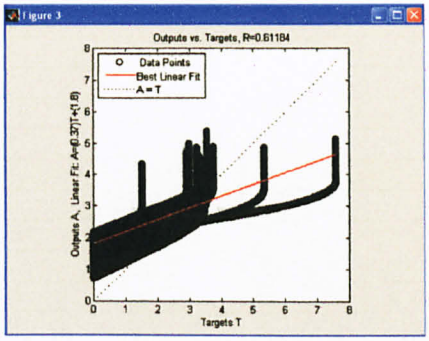


Figure 17: Post Regression for Concentration Training

The first two outputs seem to track the targets reasonably well, and the R-values are almost 0.75. The third output is not well modeled. The problem needs more work.

4.4.1.2 Temperature Training using Feedforward Backpropagation

The same algorithm is used for this training. The training stopped after 5 iterations because the validation error increased.

```
TRAINLM-calcjx, Epoch 0/100, MSE 32.5709/0, Gradient 95.4324/1e-010
TRAINLM-calcjx, Epoch 5/100, MSE 0.0278212/0, Gradient 2.66349e-012/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.
```

The result is shown in the following figure. The result here is reasonable, because the test set error and the validation set error have similar characteristics, and it does not appear that any significant overfitting has occurred.

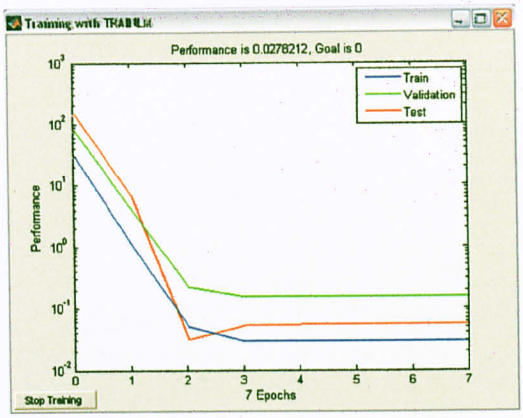


Figure 18: Temperature Training

A linear regression between the network outputs and the corresponding targets is performed. The results are shown in the following figures.

a. Testing

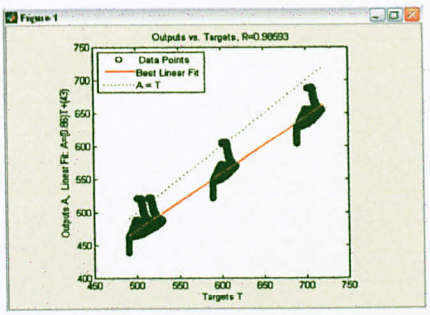


Figure 19: Post Regression for Temperature Testing

b. Validation

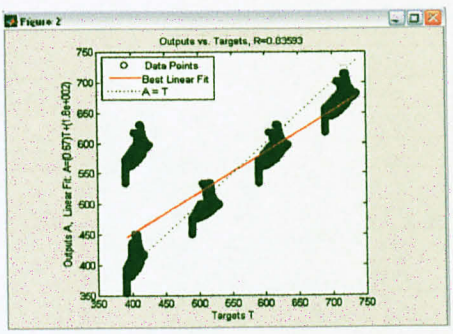


Figure 20: Post Regression for Temperature Validation

c. Training

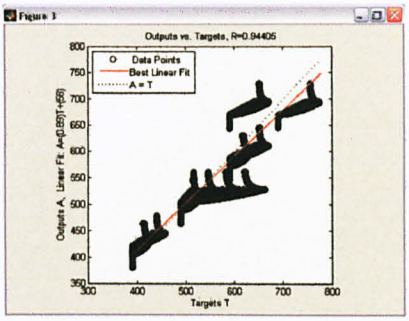


Figure 21: Post Regression for Temperature Training

4.4.1.3 Velocity Training using Feedforward Backpropagation

The same algorithm is used for this training. The training stopped after 8 iterations because the validation error increased.

```
TRAINLM-calcjx, Epoch 0/100, MSE 29.9182/0, Gradient 88.0369/1e-010
TRAINLM-calcjx, Epoch 8/100, MSE 0.251654/0, Gradient 3.13871e-009/1e-010
TRAINLM, Validation stop.
```

The result is shown in the following figure. The result here is not practical, because the test set error and the validation set error have dissimilar characteristics, and it does appear that significant overfitting has occurred.

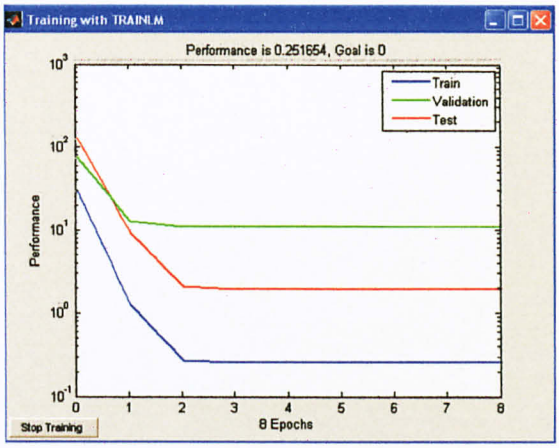


Figure 22: Velocity Training

A linear regression between the network outputs and the corresponding targets is performed. The results are shown in the following figures.

a. Testing

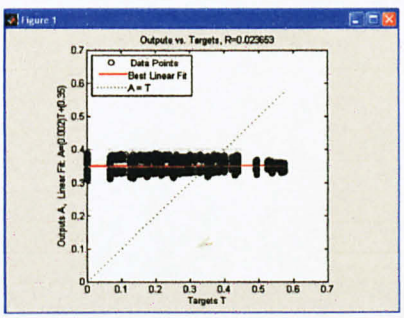


Figure 23: Post Regression for Velocity Testing

b. Validation

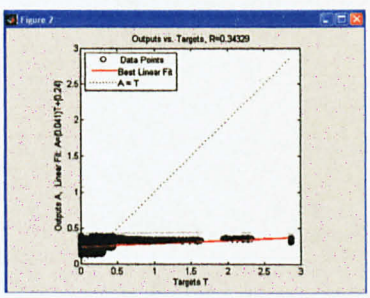


Figure 24: Post Regression for Velocity Validation

c. Training

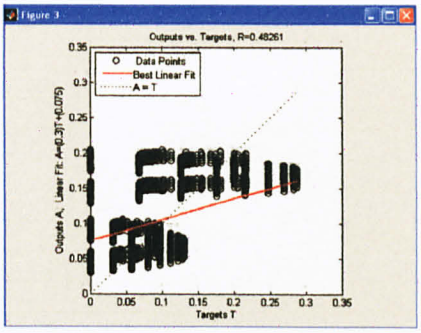


Figure 25: Post Regression for Velocity Training

All three outputs of post regression in velocity testing, validation and training was not fit the target. There is possibility that this training cannot be accurately computed based on given spectral components.

4.4.1.4 Simplified Feedforward Result

The following table summarized the post regression analysis of feed-forward backpropagation network for three identified variables.

Table 4: Post-regression Analysis for Feedforward Backpropagation Network

	Concentration	Temperature	Velocity
Slope	1.8	0.89	0.3
R	0.61184	0.94405	0.48261

From the table above, the correlation coefficient (R-value) between the outputs and targets for concentration, temperature and velocity are not perfect, which indicates a poor fit.

Apart from post-regression analysis, MATLAB has calculated the RSME and CDC of the network. Table below shows the summary of RMSE and CDC for concentration, temperature and velocity training.

Table 5: RMSE and CDC for Feed Forward Network

	Concentration	Temperature	Velocity
RMSE	0.9244	2.2219	0.1738
CDC	80.9843	0.0077	47.7730

From the table, the RMSE of temperature is very large which indicates that the output of the network deviates significantly from the targeted value as compared to the other process variables.

4.4.2 NARX Network (Nonlinear Autoregressive Network with Exogeneous Inputs Network)

NARX network is used for this training. It can be used as a predictor, to predict the next value of the input signal. It can also be used for nonlinear filtering, in which the target output is a noise-free version of the input signal. The use of the NARX network is demonstrated in another important application, the modeling of nonlinear dynamic systems.

The configuration of the NARX network is also determined from trial and error. The number of layers, neurons and training function is changed for several times. Each configuration is compared to determine the best configuration. The best configuration is summarized in table below.

Table 6: NARX Network Configuration

Parameters	Variable
Network	NARX network (newnarxsp)
Training function	TRAINLM
Adaptation learning function	LEARNGDM
Performance function	MSE

Epochs	1000
Number of layers	3
Layer 1: Number of neuron **Transfer function (PURELIN)**	5
Layer 2: Number of neuron **Transfer function (PURELIN)**	22
Layer 3: Number of neuron **Transfer Function (PURELIN)**	1

The network is trained based on this configuration. Post-regression analysis, RMSE and CDC are important characteristics to evaluate the network. The next section will discuss about post regression analysis.

4.4.2.1 Post-Training Analysis for NARX Network

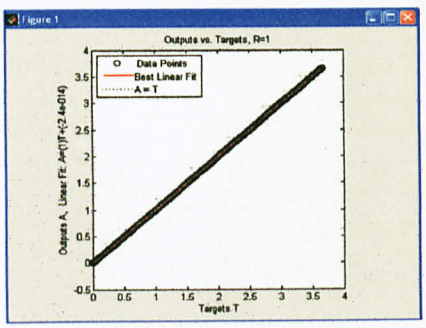


Figure 26: Post Regression for Concentration Testing

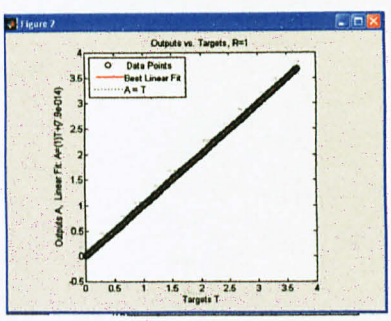


Figure 27: Post Regression for Concentration Validation

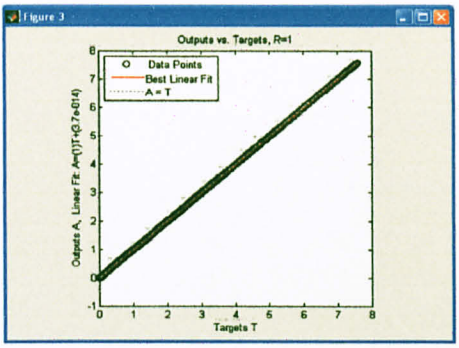


Figure 28: Post Regression for Concentration Training

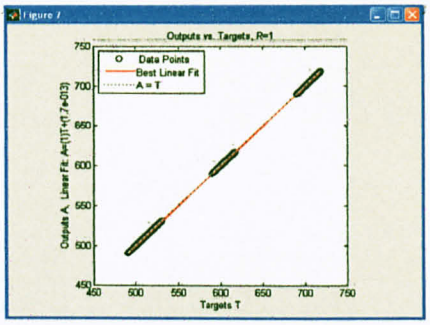


Figure 29: Post Regression for Temperature Testing

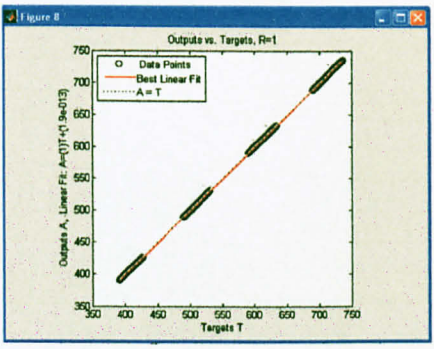


Figure 30: Post Regression for Temperature Validation

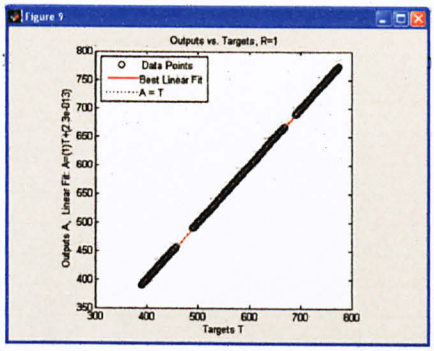


Figure 31: Post Regression for Temperature Training

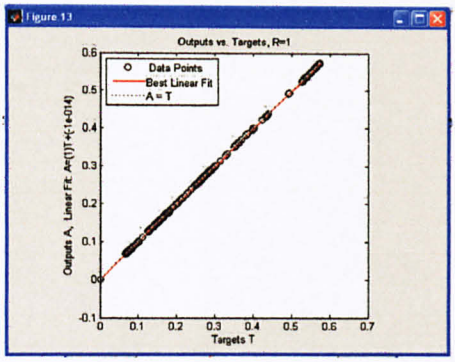


Figure 32: Post Regression for Velocity Testing

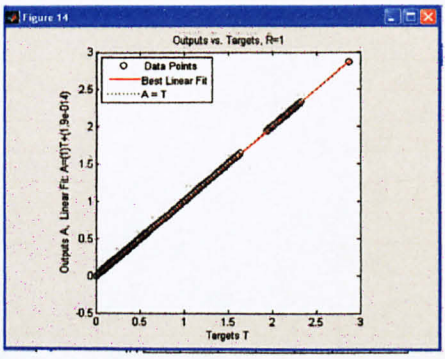


Figure 33: Post Regression for Velocity Validation

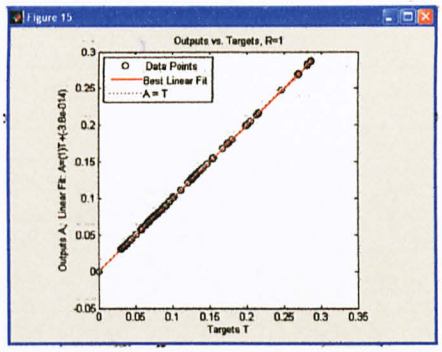


Figure 34: Post Regression for Velocity Training

The previous post regression for concentration, temperature and velocity training shows that the output fits the target very well, which is 100%, fits the experimental data. The result also shows that NARX network is the best neural network compare to Feed Forward Backpropagation network because it can predict the output for the corresponding experimental data input perfectly. For this reason, NARX network can also predict any real reactor data with 100% accuracy.

4.4.2.1 Simplified NARX Result

The following table summarized the post regression analysis of NARX network for three identified variables.

Table 7: Post-regression Analysis for NARX Network

	Concentration	Temperature	Velocity
Slope	1	1	1
R	1	1	1

From the table above, the performance of the NARX network is really good. The network is able to produce output that fits the target value. The R value shows that the variation in the output is well explained by the targets.

Apart from post-regression analysis, MATLAB has calculated the RSME and CDC of the network. Table below shows the summary of RMSE and CDC for concentration, temperature and velocity training.

Table 8: RMSE and CDC for NARX Network

	Concentration	Temperature	Velocity
RMSE	1.1946e-013	2.2798e-013	1.4848e-013
CDC	90.9076	98.7246	93.7384

4.5 Design of GUI

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. GUI is very important for the user to key in the input data and gets the predicted output in a more user friendly with very short time. For this project, the aim is to generate 2D GUI with colour map. When the user clicks the pop-up menu, the axes component displays the selected data using the specified plot. The GUI will illustrate the temperature profile, concentration profile and velocity profile of the modeled reactor. In this report, the GUI that has been developed consist pop-up menus and it display data in 2D plot. The user may click the pop-up menu and then choose the desired variables to be plotted.

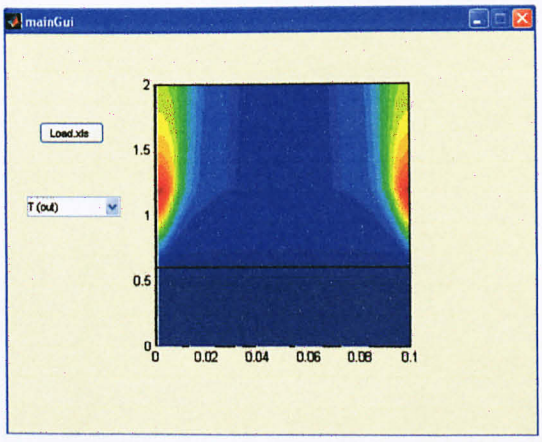


Figure 35: 2D GUI for NARX Neural Network

CHAPTER 5

CONCLUSION

A reliable experimental data has been used to simulate ethylene oxidation process in a fixed bed catalytic reactor by using COMSOL Multiphysics. The simulation generates three types of operating profile which are velocity profile, concentration profile and temperature profile. The velocity profile shows, as the fluids moving toward the center of the column, the velocity of the fluid is increasing meanwhile, the velocity of the moving fluids decreasing towards the wall region for both sides. This is happen due to the friction that exists at the wall with the moving fluids. The temperature profile shows, the temperature starts to increase toward the outlet of the reactor. This situation occurs due to the heat of reaction that has been release in this process (exothermic reactions). The concentration profile shows ethylene concentration starts to decrease toward the reactor outlet while the concentration of ethylene oxide is increasing towards the reactor outlet. The inlet reactor temperature has been varies to see the effect towards product concentration. The profile shows, the higher inlet temperature will produce more ethylene oxide product. It is due to increase in rate of reaction that has increase the number of energetic particles, thus increase the product concentration. The experimental reactor size that has been modeled is 0.1m in diameter, while the maximum simulated reactor size is 0.81m.

There are two types of neural network has been trained in this project. The first one is Feed Forward Backpropagation network and the second one is NARX network. Both neural networks have been trained using experimental data to choose the best neural network that can predict the output of corresponding input to the experimental size of reactor. The result shows that NARX network is the best neural network compare to Feed Forward Backpropagation network because it can predict the output for the corresponding experimental data input perfectly with 100% accuracy. Therefore, the NARX network can also be used to predict any data including real reactor data input from a plant. GUI has been developed for NARX network to demonstrate the performance and also predicted the output in a more user friendly.

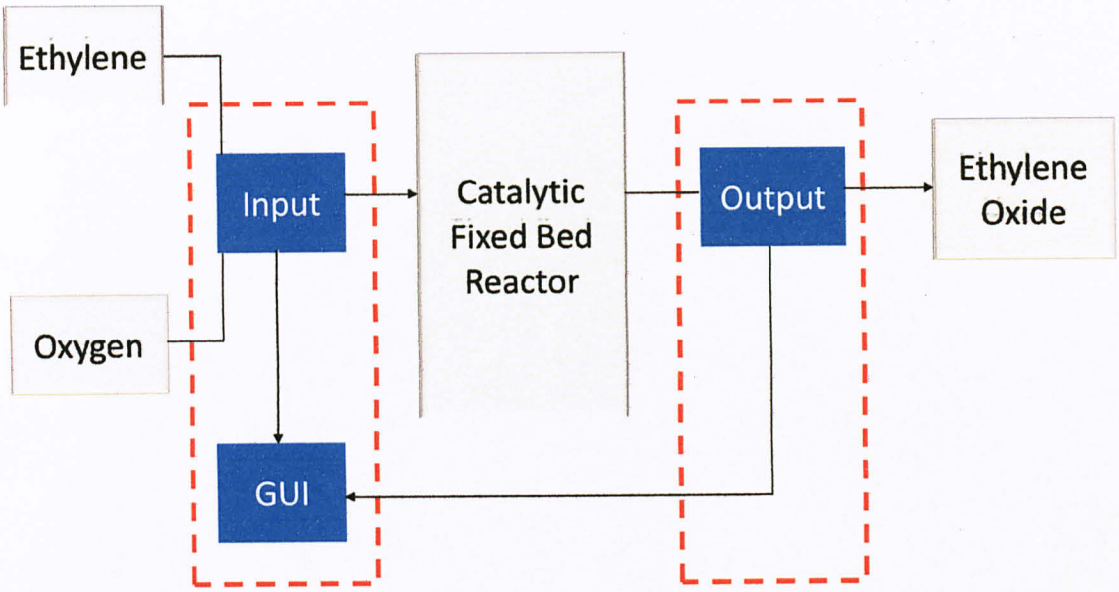


Figure 38: Design Process Flow for simulation EO in FBR

The above figure is the propose design process flow for industrial application. Ethylene and oxygen are the input to fixed bed reactor to produce ethylene oxide. The NARX network that is represented by GUI will be installed in the plant. An input signal will be sent to this system (NARX network) to execute the input data (i.e. temperature, velocity, concentration) and predict the output for the ethylene oxide process. The feedback loop will sent the output signal from the process to the system. This is to verify the system output data. If there is any problem occurs inside the reactor, it will be detected by the system accurately in a shorter time. Therefore, an immediate action can be taken before the problem get worse.

REFERENCES

1. Howard L. White, *Introduction to Industrial Chemistry,* Wiley InterScience , pg 73.
2. Edward Furimsky, *Catalyst for Upgrading Heavy Petroleum Feeds,* Elsevier, pg 44.
3. Carl Branan, *Rules of Thumb for Chemical Engineers; A manual of quick, accurate solutions to everyday process engineering problem,* GPP.
4. Walter G. Frankenburg, *Advance in Catalyst and Related Subject,* Vol. 20, Academic Press, pg 155.
5. Doug Hull. Retrieve on 24 August 2009 <http://blogs.mathworks.com/pick/2007/08/13/video-series-reading-excel-data-into-matlab-with-a-gui/ .
6. Ullmann's, *Chemical Engineering and Plant Design*, Volume 2, Wiley InterScience, pg 967,970.
7. H. Demuth & M. Beale, *Neural Network Toolbox for Use with MATLAB*, The MathWork, 1998.
8. Berty J.M, 1999, *Experiments in Catalytic Reaction Engineering: Studies in Surface Science and Catalysis*, Elsevier Science, Netherland.
9. *COMSOL- Multiphysics Chemical Engineering Module- User Guide,* 2006

APPENDICES

A. Coding for NARX Network

```
%Data
A = xlsread('Tr_C');
B = xlsread('Val_C');
C = xlsread('Te_C');

%Partitioning 40% Tr, 40% V, 20% Te
P_trc=A(:,1:5)';
P_vc=B(:,1:5)';
P_tec=C(:,1:5)';
T_trc=A(:,6)';
T_vc=B(:,6)';
T_tec=C(:,6)';

Pc_trc=con2seq(P_trc);
Tc_trc=con2seq(T_trc);
pc=[Pc_trc;Tc_trc];
tc=Tc_trc;

Pc_vc=con2seq(P_vc);
Tc_vc=con2seq(T_vc);
p1c=[Pc_vc;Tc_vc];
t1c=Tc_vc;

Pc_tec=con2seq(P_tec);
Tc_tec=con2seq(T_tec);
p2c=[Pc_tec;Tc_tec];
t2c=Tc_tec;

%set up network
d1=0;
d2=0;

netc = newnarxsp(minmax(pc),d1,d2,[5 22
1],{'purelin','purelin','purelin'});
net.trainFcn = 'trainlm';
net.trainParam.show = 10;
net.trainParam.epochs = 1000;
net.trainParam.goal   = 1e-3;

%Set up the validation and testing sets in a structure form
val.P=p1c; val.T=t1c;
test.P=p2c; test.T=t2c;
netc = train(netc,pc,tc,[],[],val,test);


%simulate network
atc =sim(netc,p2c); %Testing set
atc=cell2mat(atc);
```

```
at1c =sim(netc,p1c); %validation set
at1c=cell2mat(at1c);


at2c =sim(netc,pc); %training set
at2c=cell2mat(at2c);

%------------------graphs---------------------------

figure(1)
[slope1,intercept1,R1] = postreg(atc(1,:),T_tec(1,:)); %Testing
figure(2)
[slope2,intercept2,R2] = postreg(at1c(1,:),T_vc(1,:)); %Validation
figure(3)
[slope3,intercept3,R3] = postreg(at2c(1,:),T_trc(1,:)); %Training

figure(4) %Testing
time = 1:length(T_tec(1,:));
plot(time,T_tec(1,:),'kd-', time,atc(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Concentration
(mol/m3):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for
Concentration (mol/m3):Testing'),

figure(5) %Validation
time = 1:length(T_vc(1,:));
 plot(time,T_vc(1,:),'kd-', time,at1c(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Concentration (mol/m3):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Concentration
(mol/m3):Validation'),

figure(6) %Training
time = 1:length(T_trc(1,:));
        plot(time,T_trc(1,:),'kd-', time,at2c(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Concentration (mol/m3):Training'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Concentration
(mol/m3):Training'),

%------------------Training set performance measurement----------------
------------
```

```
% rmse calculation Training
[row1,col1] = size(T_trc);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at2c(i,j) - T_trc(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Concentration_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trc(1,i) - T_trc(1,i-1);
    b=at2c(1,i) - at2c(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_Concentration_training = (sum(D_top))*(100/(col2))

%-----------------validation set performance measurement--------------
-------------

% rmse calculation validation
[row1,col1] = size(T_vc);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1c(i,j) - T_vc(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Concentration_validation = sqrt(sum_error/col1)

% CDC calculation for validation
```

```
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_vc(1,i) - T_vc(1,i-1);
    b=at1c(1,i) - at1c(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_Concentration_validation = (sum(D_top))*(100/(col2))

%------------------testing set performance measurement-----------------
-----------

% rmse calculation testing
[row1,col1] = size(T_tec);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (atc(i,j) - T_tec(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Concentration_testing = sqrt(sum_error/col1);

% CDC calculation for testing
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_tec(1,i) - T_tec(1,i-1);
    b=atc(1,i) - atc(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
```

```
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_Concentration_testing = (sum(D_top))*(100/(col2))

%Data Temperature
D = xlsread('Tr_T');
E = xlsread('Val_T');
F = xlsread('Te_T');

%Partitioning 40% Tr, 40% V, 20% Te
P_trt=D(:,1:5)';
P_vt=E(:,1:5)';
P_tet=F(:,1:5)';
T_trt=D(:,6)';
T_vt=E(:,6)';
T_tet=F(:,6)';


Pc_trt=con2seq(P_trt);
Tc_trt=con2seq(T_trt);
pt=[Pc_trt;Tc_trt];
tt=Tc_trt;

Pc_vt=con2seq(P_vt);
Tc_vt=con2seq(T_vt);
p1t=[Pc_vt;Tc_vt];
t1t=Tc_vt;

Pc_tet=con2seq(P_tet);
Tc_tet=con2seq(T_tet);
p2t=[Pc_tet;Tc_tet];
t2t=Tc_tet;

%set up network
d1=0;
d2=0;

nett = newnarxsp(minmax(pt),d1,d2,[5 22
1],{'purelin','purelin','purelin'});
net.trainFcn = 'trainlm';
net.trainParam.show = 10;
net.trainParam.epochs = 1000;
net.trainParam.goal   = 1e-3;

%Set up the validation and testing sets in a structure form
val.P=p1t; val.T=t1t;
test.P=p2t; test.T=t2t;
nett = train(nett,pt,tt,[],[],val,test);
```

```
%simulate network
att =sim(nett,p2t); %Testing set
att=cell2mat(att);

at1t =sim(nett,p1t); %validation set
at1t=cell2mat(at1t);


at2t =sim(nett,pt); %training set
at2t=cell2mat(at2t);

%------------------graphs---------------------------

figure(7)
[slope4,intercept4,R4] = postreg(att(1,:),T_tet(1,:)); %Testing
figure(8)
[slope5,intercept5,R5] = postreg(at1t(1,:),T_vt(1,:)); %Validation
figure(9)
[slope6,intercept6,R6] = postreg(at2t(1,:),T_trt(1,:)); %Training

figure(10) %Testing
time = 1:length(T_tet(1,:));
plot(time,T_tet(1,:),'kd-', time,att(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Temperature
(K):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for
Temperature (K):Testing'),

figure(11) %Validation
time = 1:length(T_vt(1,:));
 plot(time,T_vt(1,:),'kd-', time,at1t(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Temperature
(K):Validation'),

figure(12) %Training
time = 1:length(T_trt(1,:));
        plot(time,T_trt(1,:),'kd-', time,at2t(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Training'),...
        legend('Actual','Simulated')
```

```
        Title('Actual and Simulated plot for Temperature
(K):Training'),

%------------------Training set performance measurement----------------
------------

% rmse calculation Training
[row1,col1] = size(T_trt);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at2t(i,j) - T_trt(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Temperature_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trt(1,i) - T_trt(1,i-1);
    b=at2t(1,i) - at2t(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_Temperature_training = (sum(D_top))*(100/(col2))

%-----------------validation set performance measurement--------------
-------------

% rmse calculation validation
[row1,col1] = size(T_vt);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1t(i,j) - T_vt(i,j))^2;
```

```
end
end
sum_error = sum(error_col(1,:));
rmse_Temperature_validation = sqrt(sum_error/col1)

% CDC calculation for validation
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_vt(1,i) - T_vt(1,i-1);
    b=at1t(1,i) - at1t(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_Temperature_validation = (sum(D_top))*(100/(col2))

%------------------testing set performance measurement-----------------
-----------

% rmse calculation testing
[row1,col1] = size(T_tet);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (att(i,j) - T_tet(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Temperature_testing = sqrt(sum_error/col1);

% CDC calculation for testing
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_tet(1,i) - T_tet(1,i-1);
    b=att(1,i) - att(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;
```

```
end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_Temperature_testing = (sum(D_top))*(100/(col2))

%Data velocity
G = xlsread('Tr_V');
H = xlsread('Val_V');
I = xlsread('Te_V');

%Partitioning 40% Tr, 40% V, 20% Te
P_trv=G(:,1:5)';
P_vv=H(:,1:5)';
P_tev=I(:,1:5)';
T_trv=G(:,6)';
T_vv=H(:,6)';
T_tev=I(:,6)';

Pc_trv=con2seq(P_trv);
Tc_trv=con2seq(T_trv);
pv=[Pc_trv;Tc_trv];
tv=Tc_trv;

Pc_vv=con2seq(P_vv);
Tc_vv=con2seq(T_vv);
p1v=[Pc_vv;Tc_vv];
t1v=Tc_vv;

Pc_tev=con2seq(P_tev);
Tc_tev=con2seq(T_tev);
p2v=[Pc_tev;Tc_tev];
t2v=Tc_tev;

%set up network
d1=0;
d2=0;

netv = newnarxsp(minmax(pv),d1,d2,[5 22
1],{'purelin','purelin','purelin'});
net.trainFcn = 'trainlm';
net.trainParam.show = 10;
net.trainParam.epochs = 1000;
net.trainParam.goal   = 1e-3;
```

```
%Set up the validation and testing sets in a structure form
val.P=p1v; val.T=t1v;
test.P=p2v; test.T=t2v;
netv = train(netv,pv,tv,[],[],val,test);

%simulate network
atv =sim(netv,p2v); %Testing set
atv=cell2mat(atv);

at1v =sim(netv,p1v); %validation set
at1v=cell2mat(at1v);

at2v =sim(netv,pv); %training set
at2v=cell2mat(at2v);

%-----------------graphs---------------------------

figure(13)
[slope6,intercept6,R6] = postreg(atv(1,:),T_tev(1,:)); %Testing
figure(14)
[slope7,intercept7,R7] = postreg(at1v(1,:),T_vv(1,:)); %Validation
figure(15)
[slope8,intercept8,R8] = postreg(at2v(1,:),T_trv(1,:)); %Training

figure(16) %Testing
time = 1:length(T_tev(1,:));
plot(time,T_tev(1,:),'kd-', time,atv(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Velocity (m/s):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for Velocity
(m/s):Testing'),

figure(17) %Validation
time = 1:length(T_vv(1,:));
 plot(time,T_vv(1,:),'kd-', time,at1v(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Velocity (m/s):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Velocity
(m/s):Validation'),

figure(18) %Training
time = 1:length(T_trv(1,:));
        plot(time,T_trv(1,:),'kd-', time,at2v(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Velocity (m/s):Training'),...
```

```
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Velocity (m/s):Training'),

%------------------Training set performance measurement----------------
------------

% rmse calculation Training
[row1,col1] = size(T_trv);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at2v(i,j) - T_trv(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Velocity_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trv(1,i) - T_trv(1,i-1);
    b=at2v(1,i) - at2v(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_Velocity_training = (sum(D_top))*(100/(col2))


%-----------------validation set performance measurement--------------
-------------

% rmse calculation validation
[row1,col1] = size(T_vv);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1v(i,j) - T_vv(i,j))^2;
```

```
end
end
sum_error = sum(error_col(1,:));
rmse_Velocity_validation = sqrt(sum_error/col11)

% CDC calculation for validation
d1=zeros(1,col11-1);
i=2;
for n=1:1:col11-1
    a=T_vv(1,i) - T_vv(1,i-1);
    b=at1v(1,i) - at1v(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col11-1);
m=1;
for q=1:1:col11-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_Velocity_validation = (sum(D_top))*(100/(col2))

%------------------testing set performance measurement-----------------
-----------

% rmse calculation testing
[row1,col11] = size(T_tev);
error_col = zeros(row1,col11);
for i = 1:1:row1,
for j = 1:1:col11,
    error_col(i,j) = (atv(i,j) - T_tev(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Velocity_testing = sqrt(sum_error/col11);

% CDC calculation for testing
d1=zeros(1,col11-1);
i=2;
for n=1:1:col11-1
    a=T_tev(1,i) - T_tev(1,i-1);
    b=atv(1,i) - atv(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
```

```
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_Velocity_testing = (sum(D_top))*(100/(col2))
```

B. Coding of Feedforward Network for Velocity

```
%Data
G = xlsread('Tr_V');
H = xlsread('Val_V');
I = xlsread('Te_V');

%Partitioning 40% Tr, 40% V, 20% Te
P_trv=G(:,1:5)';
P_vv=H(:,1:5)';
P_tev=I(:,1:5)';
T_trv=G(:,6)';
T_vv=H(:,6)';
T_tev=I(:,6)';

nntwarn off;

% Training set
[Pn_trv, Pmin_tr, Pmax_tr] = premnmx(P_trv);
[Tn_trv, Tmin_tr, Tmax_tr] = premnmx(T_trv);

% Validation set
[Pn_vv] = tramnmx(P_vv, Pmin_tr, Pmax_tr);
[Tn_vv] = tramnmx(T_vv, Tmin_tr, Tmax_tr);

% Testing set
[Pn_tev] = tramnmx(P_tev,Pmin_tr, Pmax_tr);
[Tn_tev] = tramnmx(T_tev, Tmin_tr, Tmax_tr);

%Setup network
netv=newff(minmax(Pn_trv), [5 30
1],{'purelin','purelin','purelin'},'trainlm','learngdm','mse');
net.trainParam.show=10;
net.trainParam.epochs=100;
net.trainparam.goal=1e-3;
net.trainparam.MU=-0.001;

% Train netwrk with early stopping
rand('seed',417000);
netv = init(netv);

% Set up the validation and testing sets in a structure form
val.P=Pn_vv; val.T=Tn_vv;
test.P=Pn_tev; test.T=Tn_tev;
[netv trv] = train(netv,Pn_trv,Tn_trv,[],[],val,test);

%Simulate network
anv = sim(netv, Pn_tev); %Testing set
atv  = postmnmx(anv, Tmin_tr, Tmax_tr); %Testing set
an1v=sim(netv,Pn_vv); %Validation
at1v=postmnmx(an1v, Tmin_tr, Tmax_tr); %Validation
an2v=sim(netv,Pn_trv); %Training
at2v=postmnmx(an2v, Tmin_tr, Tmax_tr); %Training
```

```
%------------------graphs---------------------------

figure(1)
[slope7,intercept7,R7] = postreg(atv(1,:),T_tev(1,:));
figure(2)
[slope8,intercept8,R8] = postreg(at1v(1,:),T_vv(1,:));
figure(3)
[slope9,intercept9,R9] = postreg(at2v(1,:),T_trv(1,:));

figure(4) %Testing
time = 1:length(T_tev(1,:));
plot(time,T_tev(1,:),'kd-', time,atv(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Velocity (m/s):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for Velocity
(m/s):Testing'),

figure(5) %Validation
time = 1:length(T_vv(1,:));
 plot(time,T_vv(1,:),'kd-', time,at1v(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Velocity (m/s):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Velocity
(m/s):Validation'),

figure(6) %Training
time = 1:length(T_trv(1,:));
        plot(time,T_trv(1,:),'kd-', time,at2v(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Velocity (m/s):Training'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Velocity (m/s):Training'),
%------------------Training set performance measurement----------------
------------

% rmse calculation Training
[row1,col1] = size(T_trv);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at2v(i,j) - T_trv(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
```

```
rmse_velocity_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trv(1,i) - T_trv(1,i-1);
    b=at2v(1,i) - at2v(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_velocity_training = (sum(D_top))*(100/(col2))

%-----------------validation set performance measurement--------------
-------------

% rmse calculation validation
[row1,col1] = size(T_vv);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1v(i,j) - T_vv(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_velocity_validation = sqrt(sum_error/col1)

% CDC calculation for validation
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_vv(1,i) - T_vv(1,i-1);
    b=at1v(1,i) - at1v(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end
```

```
D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_velocity_validation = (sum(D_top))*(100/(col2))

%------------------testing set performance measurement-----------------
-----------

% rmse calculation testing
[row1,col1] = size(T_tev);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (atv(i,j) - T_tev(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_velocity_testing = sqrt(sum_error/col1);

% CDC calculation for testing
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_tev(1,i) - T_tev(1,i-1);
    b=atv(1,i) - atv(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_velocity_testing = (sum(D_top))*(100/(col2))
```

C. Coding of Feedforward for Temperature

```
%Data
D = xlsread('Tr_T');
E = xlsread('Val_T');
F = xlsread('Te_T');

%Partitioning 40% Tr, 40% V, 20% Te
P_trt=D(:,1:5)';
P_vt=E(:,1:5)';
P_tet=F(:,1:5)';
T_trt=D(:,6)';
T_vt=E(:,6)';
T_tet=F(:,6)';

nntwarn off;

% Training set
[Pn_trt, Pmin_tr, Pmax_tr] = premnmx(P_trt);
[Tn_trt, Tmin_tr, Tmax_tr] = premnmx(T_trt);

% Validation set
[Pn_vt] = tramnmx(P_vt, Pmin_tr, Pmax_tr);
[Tn_vt] = tramnmx(T_vt, Tmin_tr, Tmax_tr);

% Testing set
[Pn_tet] = tramnmx(P_tet,Pmin_tr, Pmax_tr);
[Tn_tet] = tramnmx(T_tet, Tmin_tr, Tmax_tr);

%Setup network
nett=newff(minmax(Pn_trt), [5 34
1],{'purelin','purelin','purelin'},'trainlm','learngdm','mse');
net.trainParam.show=10;
net.trainParam.epochs=100;
net.trainparam.goal=1e-3;
net.trainparam.MU=-0.001;

% Train netwrk with early stopping
rand('seed',417000);
nett = init(nett);

% Set up the validation and testing sets in a structure form
val.P=Pn_vt; val.T=Tn_vt;
test.P=Pn_tet; test.T=Tn_tet;
[nett trt] = train(nett,Pn_trt,Tn_trt,[],[],val,test);

%Simulate network
ant = sim(nett, Pn_tet); %Testing set
att  = postmnmx(ant, Tmin_tr, Tmax_tr); %Testing set
an1t=sim(nett,Pn_vt); %Validation
at1t=postmnmx(an1t, Tmin_tr, Tmax_tr); %Validation
an2t=sim(nett,Pn_trt); %Training
at2t=postmnmx(an2t, Tmin_tr, Tmax_tr); %Training
```

```
%------------------graphs-----------------------------

figure(1)
[slope4,intercept4,R4] = postreg(att(1,:),T_tet(1,:)); %Testing (temp)
figure(2)
[slope5,intercept5,R5] = postreg(at1t(1,:),T_vt(1,:)); %Validation
(temp)
figure(3)
[slope6,intercept6,R6] = postreg(at2t(1,:),T_trt(1,:)); %Training
(temp)

figure(4) %Testing
time = 1:length(T_tet(1,:));
plot(time,T_tet(1,:),'kd-', time,att(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Temperature
(K):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for
Temperature (K):Testing'),

figure(5) %Validation
time = 1:length(T_vt(1,:));
 plot(time,T_vt(1,:),'kd-', time,at1t(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Temperature
(K):Validation'),

figure(6) %Training
time = 1:length(T_trt(1,:));
        plot(time,T_trt(1,:),'kd-', time,at2t(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Training'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Temperature
(K):Training'),
%------------------Training set performance measurement----------------
------------

% rmse calculation Training
[row1,col1] = size(T_trt);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at2t(i,j) - T_trt(i,j))^2;
```

```
end
end
sum_error = sum(error_col(1,:));
rmse_temperature_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trt(1,i) - T_trt(1,i-1);
    b=at2t(1,i) - at2t(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_temperature_training = (sum(D_top))*(100/(col2))
%-----------------validation set performance measurement--------------
-------------
% rmse calculation validation
[row1,col1] = size(T_vt);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1t(i,j) - T_vt(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_temperature_validation = sqrt(sum_error/col1)

% CDC calculation for validation
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_vt(1,i) - T_vt(1,i-1);
    b=at1t(1,i) - at1t(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;
```

```
end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_temperature_validation = (sum(D_top))*(100/(col2))
%-----------------testing set performance measurement----------------
----------
% rmse calculation testing
[row1,col1] = size(T_tet);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (att(i,j) - T_tet(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_temperature_testing = sqrt(sum_error/col1);

% CDC calculation for testing
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_tet(1,i) - T_tet(1,i-1);
    b=att(1,i) - att(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_temperature_testing = (sum(D_top))*(100/(col2))
```

D. Coding of Feedforward Network for Concentration

```
clc;
close all;

%Data
A = xlsread('Tr_C');
B = xlsread('Val_C');
C = xlsread('Te_C');

%Partitioning 40% Tr, 40% V, 20% Te
P_trc=A(:,1:5)';
P_vc=B(:,1:5)';
P_tec=C(:,1:5)';
T_trc=A(:,6)';
T_vc=B(:,6)';
T_tec=C(:,6)';

nntwarn off;

% Training set
[Pn_trc, Pmin_tr, Pmax_tr] = premnmx(P_trc);
[Tn_trc, Tmin_tr, Tmax_tr] = premnmx(T_trc);

% Validation set
[Pn_vc] = tramnmx(P_vc,Pmin_tr, Pmax_tr);
[Tn_vc] = tramnmx(T_vc, Tmin_tr, Tmax_tr);

% Testing set
[Pn_tec] = tramnmx(P_tec,Pmin_tr, Pmax_tr);
[Tn_tec] = tramnmx(T_tec, Tmin_tr, Tmax_tr);

%Setup network
netc=newff(minmax(Pn_trc), [5 34
1],{'purelin','purelin','purelin'},'trainlm','learngdm','mse');
net.trainParam.show=10;
net.trainParam.epochs=100;
net.trainparam.goal=1e-3;
net.trainparam.MU=-0.001;

% Train netwrk with early stopping
rand('seed',417000);
netc = init(netc);

% Set up the validation and testing sets in a structure form
val.P=Pn_vc; val.T=Tn_vc;
test.P=Pn_tec; test.T=Tn_tec;
[netc trc] = train(netc,Pn_trc,Tn_trc,[],[],val,test);

%Simulate network
anc = sim(netc, Pn_tec); %Testing set
atc  = postmnmx(anc, Tmin_tr, Tmax_tr); %Testing set
an1c=sim(netc,Pn_vc); %Validation
at1c=postmnmx(an1c, Tmin_tr, Tmax_tr); %Validation
```

```
an2c=sim(netc,Pn_trc); %Training
at2c=postmnmx(an2c, Tmin_tr, Tmax_tr); %Training


%------------------graphs---------------------------

figure(1)
[slope1,intercept1,R1] = postreg(atc(1,:),T_tec(1,:));
figure(2)
[slope2,intercept2,R2] = postreg(at1c(1,:),T_vc(1,:));
figure(3)
[slope3,intercept3,R3] = postreg(at2c(1,:),T_trc(1,:));

figure(4) %Testing
time = 1:length(T_tec(1,:));
plot(time,T_tec(1,:),'kd-', time,atc(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Concentration
():Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for Pressure
(kPa):Testing'),

figure(5) %Validation
time = 1:length(T_vc(1,:));
 plot(time,T_vc(1,:),'kd-', time,at1c(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Concentration (mol):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Concentration
(mol):Validation'),

figure(6) %Training
time = 1:length(T_trc(1,:));
        plot(time,T_trc(1,:),'kd-', time,at2c(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Concentration (mol):Training'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Concentration
(mol):Training'),

%------------------Training set performance measurement----------------
-----------

% rmse calculation Training
[row1,col1] = size(T_trc);
error_col = zeros(row1,col1);
for i = 1:1:row1,
```

```
for j = 1:1:col1,
    error_col(i,j) = (at2c(i,j) - T_trc(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Concentration_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trc(1,i) - T_trc(1,i-1);
    b=at2c(1,i) - at2c(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_Concentration_training = (sum(D_top))*(100/(col2))


%-----------------validation set performance measurement--------------
-------------

% rmse calculation validation
[row1,col1] = size(T_vc);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1c(i,j) - T_vc(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Concentration_validation = sqrt(sum_error/col1)

% CDC calculation for validation
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
```

```
    a=T_vc(1,i) - T_vc(1,i-1);
    b=atlc(1,i) - atlc(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_Concentration_validation = (sum(D_top))*(100/(col2))

%------------------testing set performance measurement----------------
-----------

% rmse calculation testing
[row1,col1] = size(T_tec);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (atc(i,j) - T_tec(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_Concentration_testing = sqrt(sum_error/col1);

% CDC calculation for testing
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_tec(1,i) - T_tec(1,i-1);
    b=atc(1,i) - atc(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
```

```
        D_top(:,m)=0;
    end

        m=m+1;
end
[row2,col2] = size(D_top);
CDC_Concentration_testing = (sum(D_top))*(100/(col2))

%Data
D = xlsread('Tr_T');
E = xlsread('Val_T');
F = xlsread('Te_T');

%Partitioning 40% Tr, 40% V, 20% Te
P_trt=D(:,1:5)';
P_vt=E(:,1:5)';
P_tet=F(:,1:5)';
T_trt=D(:,6)';
T_vt=E(:,6)';
T_tet=F(:,6)';

nntwarn off;

% Training set
[Pn_trt, Pmin_tr, Pmax_tr] = premnmx(P_trt);
[Tn_trt, Tmin_tr, Tmax_tr] = premnmx(T_trt);

% Validation set
[Pn_vt] = tramnmx(P_vt, Pmin_tr, Pmax_tr);
[Tn_vt] = tramnmx(T_vt, Tmin_tr, Tmax_tr);

% Testing set
[Pn_tet] = tramnmx(P_tet,Pmin_tr, Pmax_tr);
[Tn_tet] = tramnmx(T_tet, Tmin_tr, Tmax_tr);

%Setup network
nett=newff(minmax(Pn_trt), [5 34
1],{'purelin','purelin','purelin'},'trainlm','learngdm','mse');
net.trainParam.show=10;
net.trainParam.epochs=100;
net.trainparam.goal=1e-3;
net.trainparam.MU=-0.001;

% Train netwrk with early stopping
rand('seed',417000);
nett = init(nett);

% Set up the validation and testing sets in a structure form
val.P=Pn_vt; val.T=Tn_vt;
test.P=Pn_tet; test.T=Tn_tet;
[nett trt] = train(nett,Pn_trt,Tn_trt,[],[],val,test);

%Simulate network
ant = sim(nett, Pn_tet); %Testing set
att  = postmnmx(ant, Tmin_tr, Tmax_tr); %Testing set
```

```
an1t=sim(nett,Pn_vt); %Validation
at1t=postmnmx(an1t, Tmin_tr, Tmax_tr); %Validation
an2t=sim(nett,Pn_trt); %Training
at2t=postmnmx(an2t, Tmin_tr, Tmax_tr); %Training


%------------------graphs---------------------------

figure(1)
[slope4,intercept4,R4] = postreg(att(1,:),T_tet(1,:)); %Testing (temp)
figure(2)
[slope5,intercept5,R5] = postreg(at1t(1,:),T_vt(1,:)); %Validation
(temp)
figure(3)
[slope6,intercept6,R6] = postreg(at2t(1,:),T_trt(1,:)); %Training
(temp)

figure(4) %Testing
time = 1:length(T_tet(1,:));
plot(time,T_tet(1,:),'kd-', time,att(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Temperature
(K):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for
Temperature (K):Testing'),

figure(5) %Validation
time = 1:length(T_vt(1,:));
 plot(time,T_vt(1,:),'kd-', time,at1t(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Temperature
(K):Validation'),

figure(6) %Training
time = 1:length(T_trt(1,:));
        plot(time,T_trt(1,:),'kd-', time,at2t(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Training'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Temperature
(K):Training'),

%------------------Training set performance measurement----------------
------------
```

```
an1t=sim(nett,Pn_vt); %Validation
at1t=postmnmx(an1t, Tmin_tr, Tmax_tr); %Validation
an2t=sim(nett,Pn_trt); %Training
at2t=postmnmx(an2t, Tmin_tr, Tmax_tr); %Training


%-----------------graphs---------------------------

figure(1)
[slope4,intercept4,R4] = postreg(att(1,:),T_tet(1,:)); %Testing (temp)
figure(2)
[slope5,intercept5,R5] = postreg(at1t(1,:),T_vt(1,:)); %Validation
(temp)
figure(3)
[slope6,intercept6,R6] = postreg(at2t(1,:),T_trt(1,:)); %Training
(temp)

figure(4) %Testing
time = 1:length(T_tet(1,:));
plot(time,T_tet(1,:),'kd-', time,att(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Temperature
(K):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for
Temperature (K):Testing'),

figure(5) %Validation
time = 1:length(T_vt(1,:));
 plot(time,T_vt(1,:),'kd-', time,at1t(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Temperature
(K):Validation'),

figure(6) %Training
time = 1:length(T_trt(1,:));
        plot(time,T_trt(1,:),'kd-', time,at2t(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Temperature (K):Training'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Temperature
(K):Training'),

%-----------------Training set performance measurement----------------
-----------
```

```
% rmse calculation Training
[row1,col1] = size(T_trt);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at2t(i,j) - T_trt(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_temperature_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trt(1,i) - T_trt(1,i-1);
    b=at2t(1,i) - at2t(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_temperature_training = (sum(D_top))*(100/(col2))


%-----------------validation set performance measurement--------------
-------------

% rmse calculation validation
[row1,col1] = size(T_vt);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1t(i,j) - T_vt(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_temperature_validation = sqrt(sum_error/col1)
```

```
% CDC calculation for validation
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_vt(1,i) - T_vt(1,i-1);
    b=at1t(1,i) - at1t(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_temperature_validation = (sum(D_top))*(100/(col2))

%------------------testing set performance measurement----------------
-----------

% rmse calculation testing
[row1,col1] = size(T_tet);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (att(i,j) - T_tet(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_temperature_testing = sqrt(sum_error/col1);

% CDC calculation for testing
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_tet(1,i) - T_tet(1,i-1);
    b=att(1,i) - att(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
```

```
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_temperature_testing = (sum(D_top))*(100/(col2))

%Data
G = xlsread('Tr_V');
H = xlsread('Val_V');
I = xlsread('Te_V');


%Partitioning 40% Tr, 40% V, 20% Te
P_trv=G(:,1:5)';
P_vv=H(:,1:5)';
P_tev=I(:,1:5)';
T_trv=G(:,6)';
T_vv=H(:,6)';
T_tev=I(:,6)';

nntwarn off;

% Training set
[Pn_trv, Pmin_tr, Pmax_tr] = premnmx(P_trv);
[Tn_trv, Tmin_tr, Tmax_tr] = premnmx(T_trv);

% Validation set
[Pn_vv] = tramnmx(P_vv, Pmin_tr, Pmax_tr);
[Tn_vv] = tramnmx(T_vv, Tmin_tr, Tmax_tr);

% Testing set
[Pn_tev] = tramnmx(P_tev, Pmin_tr, Pmax_tr);
[Tn_tev] = tramnmx(T_tev, Tmin_tr, Tmax_tr);

%Setup network
netv=newff(minmax(Pn_trv), [5 34
1],{'purelin','purelin','purelin'},'trainlm','learngdm','mse');
net.trainParam.show=10;
net.trainParam.epochs=100;
net.trainparam.goal=1e-3;
net.trainparam.MU=-0.001;

% Train netwrk with early stopping
rand('seed',417000);
netv = init(netv);

% Set up the validation and testing sets in a structure form
val.P=Pn_vv; val.T=Tn_vv;
test.P=Pn_tev; test.T=Tn_tev;
```

```
[netv trv] = train(netv,Pn_trv,Tn_trv,[],[],val,test);

%Simulate network
anv = sim(netv, Pn_tev); %Testing set
atv  = postmnmx(anv, Tmin_tr, Tmax_tr); %Testing set
an1v=sim(netv,Pn_vv); %Validation
at1v=postmnmx(an1v, Tmin_tr, Tmax_tr); %Validation
an2v=sim(netv,Pn_trv); %Training
at2v=postmnmx(an2v, Tmin_tr, Tmax_tr); %Training


%------------------graphs----------------------------

figure(1)
[slope7,intercept7,R7] = postreg(atv(1,:),T_tev(1,:));
figure(2)
[slope8,intercept8,R8] = postreg(at1v(1,:),T_vv(1,:));
figure(3)
[slope9,intercept9,R9] = postreg(at2v(1,:),T_trv(1,:));

figure(4) %Testing
time = 1:length(T_tev(1,:));
plot(time,T_tev(1,:),'kd-', time,atv(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
                    xlabel('Time'), ylabel('Velocity
(BV/hr):Testing'),...
                        legend('Actual','Simulated')
                         Title('Actual and Simulated plot for Velocity
(BV/hr):Testing'),

figure(5) %Validation
time = 1:length(T_vv(1,:));
 plot(time,T_vv(1,:),'kd-', time,at1v(1,:),'r-','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Velocity (BV/hr):Validation'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Velocity
(BV/hr):Validation'),

figure(6) %Training
time = 1:length(T_trv(1,:));
       plot(time,T_trv(1,:),'kd-', time,at2v(1,:),'r-
','LineWidth',0.5,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',4);
        xlabel('Time'), ylabel('Velocity (BV/hr):Training'),...
        legend('Actual','Simulated')
        Title('Actual and Simulated plot for Velocity
(BV/hr):Training'),
```

```
%-----------------Training set performance measurement---------------
-----------

% rmse calculation Training
[row1,col1] = size(T_trv);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at2v(i,j) - T_trv(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_velocity_training = sqrt(sum_error/col1)

% CDC calculation for training
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_trv(1,i) - T_trv(1,i-1);
    b=at2v(1,i) - at2v(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end

[row2,col2] = size(D_top);
CDC_velocity_training = (sum(D_top))*(100/(col2))


%-----------------validation set performance measurement--------------
-------------

% rmse calculation validation
[row1,col1] = size(T_vv);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (at1v(i,j) - T_vv(i,j))^2;

end
end
```

```
sum_error = sum(error_col(1,:));
rmse_velocity_validation = sqrt(sum_error/col1)

% CDC calculation for validation
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_vv(1,i) - T_vv(1,i-1);
    b=at1v(1,i) - at1v(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end

D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_velocity_validation = (sum(D_top))*(100/(col2))

%------------------testing set performance measurement----------------
-----------

% rmse calculation testing
[row1,col1] = size(T_tev);
error_col = zeros(row1,col1);
for i = 1:1:row1,
for j = 1:1:col1,
    error_col(i,j) = (atv(i,j) - T_tev(i,j))^2;

end
end
sum_error = sum(error_col(1,:));
rmse_velocity_testing = sqrt(sum_error/col1);

% CDC calculation for testing
d1=zeros(1,col1-1);
i=2;
for n=1:1:col1-1
    a=T_tev(1,i) - T_tev(1,i-1);
    b=atv(1,i) - atv(1,i-1);
    c=a*b;
    d1(:,i-1)=c;
        i=i+1;

end
```

```
D_top=zeros(1,col1-1);
m=1;
for q=1:1:col1-1
    if d1(:,m)>0
         D_top(:,m)=1;
    else
        D_top(:,m)=0;
    end

         m=m+1;
end
[row2,col2] = size(D_top);
CDC_velocity_testing = (sum(D_top))*(100/(col2))
```

E. Coding for GUI Development

```
function varargout = mainGui(varargin)
% MAINGUI M-file for mainGui.fig
%      MAINGUI, by itself, creates a new MAINGUI or raises the existing
%      singleton*.
%
%      H = MAINGUI returns the handle to a new MAINGUI or the handle to
%      the existing singleton*.
%
%      MAINGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MAINGUI.M with the given input
arguments.
%
%      MAINGUI('Property','Value',...) creates a new MAINGUI or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before mainGui_OpeningFunction gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to mainGui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mainGui

% Last Modified by GUIDE v2.5 23-Jul-2009 22:18:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mainGui_OpeningFcn, ...
                   'gui_OutputFcn',  @mainGui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before mainGui is made visible.
function mainGui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mainGui (see VARARGIN)

% Choose default command line output for mainGui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mainGui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = mainGui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenuX.
function popupmenuX_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenuX contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenuX


% --- Executes during object creation, after setting all properties.
function popupmenuX_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in popupmenuY.
function popupmenuY_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenuY contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenuY


% --- Executes during object creation, after setting all properties.
function popupmenuY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbuttonLoadXLS.
function pushbuttonLoadXLS_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonLoadXLS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.fileName = uigetfile('*.xls')
guidata(hObject, handles)
setPopupmenuString(handles.popupmenuX, eventdata, handles)
setPopupmenuString(handles.popupmenuY, eventdata, handles)
set(handles.popupmenuX,'callback','mainGui(''updateAxes'',gcbo,[],guida
ta(gcbo))')
set(handles.popupmenuY,'callback','mainGui(''updateAxes'',gcbo,[],guida
ta(gcbo))')

function setPopupmenuString(hObject, eventdata, handles)

fileName = handles.fileName;
[numbers, colNames] = xlsread(fileName);
set(hObject,'string',colNames);

function [x,y] = readExcelColumns(fileName, xColNum, yColNum)


a = xlsread(fileName);
```

```
x = a(:,xColNum);
y = a(:,yColNum);

function updateAxes(hObject, eventdata, handles)

xColNum  = get(handles.popupmenuX, 'value');
yColNum  = get(handles.popupmenuY, 'value');
fileName = handles.fileName;

[x,y] = readExcelColumns(fileName, xColNum, yColNum)

plot(handles.axes1, x,y)
```